НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.582

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмний метод оптимізації та формалізації процесу розроблення інтерфейсів для веб-застосунків»

Виконав:
студент ІІ курсу, групи КП-81мп
Царіков Максим Сергійович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н. доцент,
Олещенко Любов Михайлівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри АУТС ФІОТ, к.т.н., доцент,
Полторак Вадим Петрович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (освітня програма) – 121 «Інженерія програмного забезпечення» ("Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем")

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Царікову Максиму Сергійовичу

1. Тема дисертації «Програмний метод оптимізації та формалізації процесу розроблення інтерфейсів для веб-застосунків», науковий керівник дисертації доцент кафедри ПЗКС, к.т.н., Олещенко Любов Михайлівна затверджені наказом по університету від «13» листопада 2019 р. № 3895-с.

2. Термін подання студентом дисертації «14» грудня 2019 р.

3. Об'єкт дослідження: процес створення інтерфейсу для веб-застосунків.

4. Предмет дослідження: методи використання часової метрики користувацьких дій для оптимізації розташування UI елементів в інтерфейсах веб-застосунків.

5. Перелік завдань, які потрібно розробити:
- створення програмного модуля, що буде швидко встановлюватися у застосунках, що використовують React.JS;
- створення алгоритму оцінки UI елементів на основі метрики користувацьких дій;
- виконати програмну реалізацію розробленого методу;
- виконати порівняння запропонованого методу із існуючими аналогами та надати оцінку отриманих результатів;

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
- загальна архітектура підходу;
- схема організації крос-платформного тексту програми у розроблюваній системі;
- діаграма класів каналу комунікації;
- результати аналізу розробленого програмного забезпечення;
- дерево проблем та рішень.

7. Орієнтовний перелік публікацій:

– XII наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» (ПМК-2019);

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., к.т.н., доцент		

9. Дата видачі завдання «25» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.11.2018	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.12.2018	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	15.02.2019	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	05.04.2019	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.05.2019	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	15.06.2019	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК-2019	13.11.2019	
8.	Оформлення текстової і графічної частини магістерської дисертації	05.12.2019	

Студент М.С. Царіков

Науковий керівник дисертації Л.М. Олещенко

РЕФЕРАТ

Актуальність. В двадцять першому столітті з активним розвитком інформаційних технологій розпочався і новий етап розвитку бізнесу. На сьогодні майже будь-яка комерційна організація має змогу реалізовувати свої товари і послуги через мережу інтернет. Це в свою чергу провокує появу нових веб-сайтів, і, як наслідок, високу конкуренцію серед них.

В Україні, як і в світі, щороку створюється все більше веб-сервісів, що мають комерційний характер. Інтерфейси цих веб-сервісів загалом мають багато спільного. Найчастіше, вони наслідують форму та дизайн від тих, що вже мають великий успіх. Однак кожен бізнес прагне до унікальності, і постійно намагається створити щось нове, і це не завжди вдається, через те, що інтерфейс, його форма та дизайн є поняттям більш естетичним на даний момент. Оцінкою його якості та успіху займаються окремі працівники. Їх робота на сьогодні є не систематизованою та не плідною. Саме тому в сфері інтернет технологій створилася окрема низка додатків, що оцінюють продуктивність роботи того чи іншого веб-сайту. Основним недоліком таких додатків є те, що вони надають велику кількість статистичних даних, але не надають висновків по ній.

В даній роботі було запропоновано рішення даної проблеми, а саме програмний метод, який на базі даних користувацької статистики буде вказувати на чіткі проблеми в інтерфейсі веб-додатків. Даний метод є спробою формалізувати процес підтримки інтерфейсу веб-додатків, що надає змогу суттєво покращити основні користувацькі метрики веб-сайту і як наслідок реалізацію продукції.

Об’єктом дослідження в даній роботі є процес створення інтерфейсу для веб-застосунків.

Предметом дослідження є методи використання часової метрики користувацьких дій для оптимізації розташування UI елементів в інтерфейсах веб-застосунків.

Метою дослідження є формалізувати процес розроблення інтерфейсу веб-застосунків, створити програмний метод, що на основі користувацької активності буде повноцінно відображати недоліки в проектуванні інтерфейсу веб-застосунку.

Методи дослідження. В роботі використовуються методи комп'ютерного моделювання, статистичні та емпіричні методи.

Наукова новизна роботи полягає в наступному.

Запропоновано новий метод оцінки "важливості" елементів інтерфейсу веб-сторінки, що надає можливість формалізувати процес підтримки веб-сайтів, а також внесення змін в зовнішній вигляд веб-сторінок.

Практична цінність отриманих в роботі результатів полягає в тому, що запропонований метод надає не тільки статистичну інформації по веб-сайту та кожній його сторінці, але й статистику по кожному елементу інтерфейсу веб-сайту, що знижує час аналізу такої інформації, та надає чіткі вказівки на зміни, які можуть покращити користувацьку статистику веб-застосунку.

Апробація роботи. Основні положення і результати роботи доповідалися та обговорювалися на XII науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2019.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, п'яти розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, наведено відомості про апробацію результатів.

У першому розділі сформульовано мету і задачі дослідження; розглянуто сучасні методи та додатки для збору користувацької статистики; наведено їх переваги та недоліки; розглянуто наявні комерційні програмні продукти, що надають статистику користувацьких дій по веб-застосунку.

У другому розділі було запропоновано метод аналізу збору користувацьких даних, визначено алгоритми і структури даних, що необхідні для реалізації запропонованого модифікованого методу.

У третьому розділі описано архітектуру програмного забезпечення, що була використана під-час розробки програмного забезпечення; визначено кінцевий алгоритм та критерії, по яким аналізується користувацька статистика, що реалізує запропонований метод; описано формат вхідних даних, з якими буде працювати реалізована система.

У четвертому розділі описано методику оцінювання ефективності методів безперервної передачі даних; наведено критерії визначення ефективності запропонованого методу; наведено результати тестування алгоритму, розробленого на основі запропонованого методу; запропоновано шляхи подальшого вдосконалення розробленого методу.

У п’ятому розділі побудовано бізнес-модель, яка обґрунтовує доцільність реалізованого програмного забезпечення та прогнозує його потенційну прибутковість у майбутньому.

У висновках проаналізовано отримані результати роботи.

У додатках наведена копія презентації.

Робота виконана на 71 аркушах, містить 2 додатки та посилання на список використаних літературних джерел з 24 найменувань. У роботі наведено 11 рисунків, 7 таблиць та 5 лістингів.

Ключові слова: UI, User Metrics, User Drop Rate, UAST.

ABSTRACT

Topicality. In the twenty-first century, with the active development of information technology, a new stage of business development began. Today, almost any commercial organization is able to market its products and services through the Internet. This in turn provokes the emergence of new websites and, as a result, high competition among them.

In Ukraine, as in the world, every year more and more commercial services are being created. The interfaces of these web services have a lot in common. Most often, they inherit form and design from those who are already having great success. However, every business strives for uniqueness, and constantly tries to create something new, and this is not always possible, because the interface, its form and design is a concept more aesthetic at the moment. Individual employees evaluate their quality and success. Their work today is not systematic or fruitful. That is why in the field of Internet technologies, a separate set of applications has been created that assess the performance of a website. The main disadvantage of such applications is that they provide a large amount of statistics, but do not provide conclusions on it.

This paper proposes a solution to this problem, namely a software method that, based on user statistics, will indicate clear problems in the Web application interface. This method is an attempt to formalize the process of supporting the Web application interface, which can significantly improve the basic user metrics of the website and as a result of sales.

The object of research in this paper is the process of creating an interface for web applications.

The subject of the study are methods of using a custom action timeline to optimize the location of UI elements in web application interfaces.

The aim of the study is to formalize the process of developing a web application interface, to create a programming method that, based on user

activity, will fully reflect the deficiencies in the design of the web application interface.

Research methods. Methods of computer modelling, statistical and empirical methods are used in this work.

The scientific novelty. A new method of assessing the "importance" of elements of a web page interface is offered, which makes it possible to formalize the process of maintaining web sites and to make changes to the appearance of web pages.

The practical value of the results obtained in the work is that the proposed method provides not only statistical information on the website and each page, but also statistics on each element of the website interface, which reduces the time of analysis of such information, and provides clear guidance on changes that can improve the user experience. web application statistics.

Test work. The main provisions and results of work were reported and discussed at the XIIth International Conference of Masters and Postgraduate Students "Applied Mathematics and Computer", PMK-2019.

Structure and scope of work. The master's dissertation consists of an introduction, five sections, conclusions and appendices.

The introduction provides a general description of the work, assesses the current state of the problem, substantiates the relevance of the research direction, provides information on the validation of results.

The first section sets out the research goals and objectives; modern methods and applications for collecting user statistics are reviewed; their advantages and disadvantages are given; Existing commercial software products that provide statistics for user actions on a web application are reviewed.

The second section proposes a method for analyzing user data collection, identifies the algorithms and data structures required to implement the proposed modified method.

The third section describes the software architecture that was used during the software development; determined the final algorithm and criteria by which

the user statistics are analyzed that implements the proposed method; describes the format of the input data that the system will work with.

The fourth section describes how to evaluate the effectiveness of continuous data transmission methods; the criteria for determining the effectiveness of the proposed method are given; the results of testing an algorithm developed on the basis of the proposed method; ways of further improvement of the developed method are offered.

The fifth section builds a business model that substantiates the feasibility of the software being implemented and predicts its potential profitability in the future.

The conclusions are analyzed the results of work.

The attachments contain a copy of the presentation.

The work is done on 71 sheets, contains 2 supplements and a link to the list of used literary sources of 24 titles. The paper presents 11 figures, 7 tables and 5 listings.

Keywords: UI, User Metrics, User Drop Rate, UAST.

РЕФЕРАТ

Актуальность. В двадцать первом веке с активным развитием информационных технологий начался и новый этап развития бизнеса. На сегодня почти любая коммерческая организация имеет возможность реализовывать свои товары и услуги через интернет. Это в свою очередь провоцирует появление новых веб-сайтов, и, как следствие, высокой конкуренции среди них.

В Украине, как и в мире, ежегодно создается все больше веб-сервисов, имеющих коммерческий характер. Интерфейсы этих веб-сервисов в целом имеют много общего. Чаще всего, они подражают форму и дизайн от тех, которые уже имеют большой успех. Однако каждый бизнес стремится к уникальности, и постоянно пытается создать что-то новое, и это не всегда удается, потому, что интерфейс, его форма и дизайн является понятием более эстетичным на данный момент.

Оценкой его качества и успеха занимаются отдельные работники. Их работа на сегодня не систематизированной но не плодотворной. Именно поэтому в сфере интернет технологий создалась отдельная ряд приложений, оценивают производительность работы того или иного сайта. Основным недостатком таких приложений является то, что они предоставляют большое количество статистических данных, но не предоставляют выводов по ней.

В данной работе было предложено решение данной проблемы, а именно программный метод, который в базе данных пользовательской статистики будет указывать на четкие проблемы в интерфейсе веб-приложений. Данный метод является попыткой формализовать процесс поддержки интерфейса веб-приложений, дает возможность существенно улучшить основные пользовательские метрики сайта и как следствие реализацию продукции.

Объектом исследования в данной работе является процесс создания интерфейса для веб-приложений.

Предметом исследования являются методы использования временной метрики пользовательских действий для оптимизации расположения UI элементов в интерфейсах веб-приложений.

Целью исследования является формализовать процесс разработки интерфейса веб-приложений, создать программный метод, на основе пользовательской активности будет полноценно отражать недостатки в проектировании интерфейса веб-приложения.

Методы исследования. В работе используются методы компьютерного моделирования, статистические и эмпирические методы.

Научная новизна работы заключается в следующем.

Предложен новый метод оценки "важности" элементов интерфейса веб-страницы, позволяет формализовать процесс поддержки веб-сайтов, а также внесение изменений во внешний вид веб-страниц.

Практическая ценность полученных в работе результатов заключается в том, что предложенный метод оказывает не только статистическую информации по веб-сайта и каждой его странице, но и статистику по каждому элементу интерфейса веб-сайта, снижает время анализа такой информации, и предоставляет четкие указания на изменения, которые могут улучшить пользовательскую статистику веб-приложения.

Апробация работы. Основные положения и результаты работы докладывались и обсуждались на XII научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2019.

Структура и объем работы. Магистерская диссертация состоит из введения, пяти глав, заключения и приложений.

Во введении дано общая характеристика работы, выполнена оценка современного состояния проблемы, обоснована актуальность направления исследований, приведены сведения об апробации результатов.

В первой главе сформулированы цель и задачи исследования; рассмотрены современные методы и приложения для сбора

пользовательской статистики; приведены их преимущества и недостатки; рассмотрены имеющиеся коммерческие программные продукты, предоставляющие статистику пользовательских действий по веб-приложения.

Во втором разделе был предложен метод анализа сбора пользовательских данных, определены алгоритмы и структуры данных, необходимых для реализации предложенного модифицированного метода.

В третьем разделе описано архитектуру программного обеспечения, которая была использована во-время разработки программного обеспечения; определены конечный алгоритм и критерии, по которым анализируется пользовательская статистика, реализует предложенный метод, описан формат входных данных, с которыми будет работать реализованная система.

В четвертом разделе описана методика оценки эффективности методов непрерывной передачи данных приведены критерии определения эффективности предложенного метода; приведены результаты тестирования алгоритма, разработанного на основе предложенного метода; предложены пути дальнейшего совершенствования разработанного метода.

В пятом разделе построено бизнес-модель, которая обосновывает целесообразность реализованного программного обеспечения прогнозирует его потенциальную прибыльность в будущем.

В выводах проанализированы полученные результаты работы.

В приложениях приведена копия презентации.

Работа выполнена на 71 листах, содержит 2 приложения и ссылки на список использованных литературных источников из 24 наименований. В работе приведены 11 рисунков, 7 таблиц и 5 листингов.

Ключевые слова: UI, User Metrics, User Drop Rate, UAST.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

React JS – фреймворк для створення веб-застосунків.

User time metrics – параметр оцінки користувацьких дій у різні проміжки часу.

User drop rate – параметр оцінки окремих веб-сторінок, що показує величину користувачів, які припинили сесію на даному веб-застосунку.

WLAN (Wireless Local Area Network) – бездротова локальна мережа.

UI (User Interface) – користувацький інтерфейс.

UAST (User Average Session Time) – середній час сесії користувача на веб-сайті.

ВСТУП

З розвитком інформаційних технологій, та особливо з розвитком мережі Інтернет, ці технології почали впливати та ставати частиною багатьох сфер діяльності. На сьогодні велика кількість комерційних операцій виконується з використанням світової мережі. Товари і послуги майже будь-якого характеру на сьогодні можуть бути реалізовані онлайн. Саме тому майже кожен магазин у світі має свій веб-сайт, що, в свою чергу, є механізмом взаємодії з користувачем.

Таким чином, веб-сайт бізнесу можна вважати його візитною карткою, і саме тому велика увага приділяється його зовнішньому оформленню. Зазвичай дизайн та форму інтерфейсу власники нових веб-сайтів намагаються наслідувати від тих, що вже мають успіх. Це спричиняє те, що за своєю структурою більшість інтернет-магазинів схожі, однак водночас кожен прагне до певної унікальності.

При створенні комерційного веб-сервісу дуже велика увага приділяється його зовнішньому вигляду. Але на сьогодні питання створення інтерфейсу (його форми) є питанням більш естетичним, а значить, є складним для подальшого аналізу. Саме тому в даній роботі було вирішено створити програмне забезпечення, що на базі запропонованого методу буде аналізувати продуктивність створеного інтерфейсу веб-сайту.

Оскільки з появою комерційних веб-сервісів почали з’являтися сервіси, що націлені на покращення роботу перших, на сьогодні існують аналоги, що надають велику кількість користувацької статистики по кожній сторінці веб-сайту. Проте жоден з них не пропонує висновків з зібраної метрики. Саме тому було вирішено створити метод, що буде не лише аналізувати користувацькі дії, але й вказувати на те, які можливі помилки були здійсненні при створенні інтерфейсу.

1. АНАЛІЗ ІСНУЮЧИХ ІСНУЮЧИХ РІШЕНЬ

1.1. Огляд існуючих комерційних програмних продуктів

На сьогодні існує велика кількість додатків, що збирають користувацьку статистику на встановленому веб-сайті. Серед них виділяється Google Analytics. В своїй реалізації він багато в чому покладається на вже створений продукт: пошукову систему Google, а також Google API. За рахунок цього цей додаток має найширший вибір користувацьких метрик які він може відстежувати.

Усі інші додатки, що наразі існують у світі, створені за іншим принципом і не мають тих можливостей, які має Google Analytics. Недоліком створення таких додатків є те, що такі додатки мають створюватися окремо під кожну платформу. Тобто для створення додатку, який можна встановити на веб-сайті, що був створений за допомогою CMS WordPress або іншої CMS, або був вручну написаний за допомогою засобів javascript, python, Java або інших засобів чи платформ, то під кожну з цих платформ потрібно створювати новий додаток. В даній роботі було вирішено створити додаток, що можна встановити на будь-який сайт, створений за допомогою react.js. Більше про існуючі аналоги далі.

1.1.1. *Google Analytics*

Google Analytics може збирати дані різними способами, найчастіше – це використання коду відстеження Google Analytics, і це найчастіше реалізовано за допомогою менеджера тегів Google, тому даний додаток фактично використовує диспетчер тегів для розгортання коду відстеження на обраному веб-сайті. Даний додаток також може збирати дані з HTML або прискорених мобільних сторінок, а також мобільних додатків. Наступним етапом є обробка даних. Це конфігурація, що застосовується в додатку, для відстеження інформації, яку користувач хоче отримати.

Також даний додаток має звітність. Тут ми можемо отримати доступ до цих даних через веб-інтерфейс або за допомогою API Google Analytics, ми також можемо використовувати інші інструменти, наприклад, аркуші Google Data Studio та інше. Отже архітектуру додатку можна описати чотирма ключовими компонентами (рис. 1):

- модуль збору даних;
- модуль обробки даних;
- модуль конфігурації пошуку даних;
- модуль відображення користувацької статистики.

Додаток має можливості налаштування способу подання звітів, використовуючи параметри, вбудовані в Google Analytics, включаючи спеціальні звіти та інформаційні панелі.

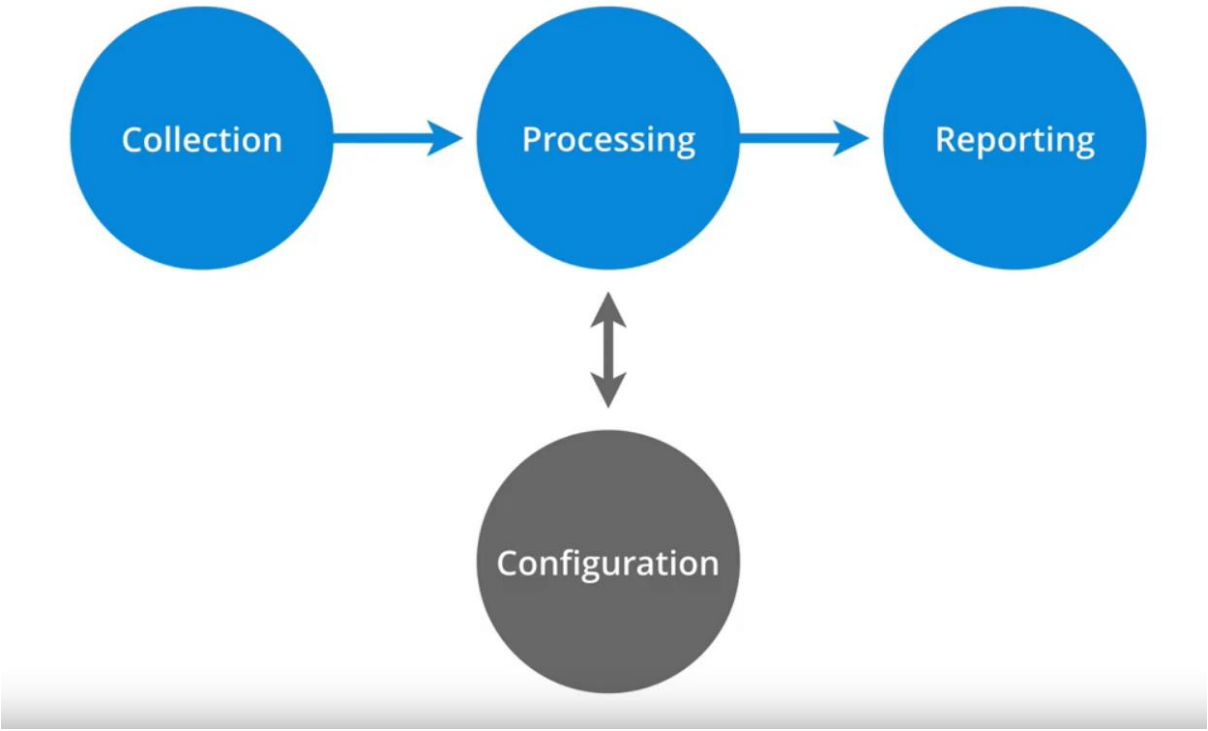


Рис. 1. Основні функції Google Analytics

Більша частина функціональних можливостей, що стосується збору даних, міститися в диспетчері тегів Google. Його потрібно розміщувати на всіх різних сторінках сайту.

Google Analytics відстежує велику кількість різних користувацьких показників, що можна по різному класифікувати. На рис. 2 показано меню з головними показниками, які стосуються загальної користувацької активності на всьому веб-сайті.

Однією з головних метрик даного додатку є кількість відвідувань сайту. Це кількість користувачів, які відвідали веб-сайт. Тому, наприклад, тут ми можемо побачити “органічний пошук”: це користувачі, які потрапили на сайт через пошукову систему Google, і ми можемо побачити кількість сеансів, тобто кількість відвідувань.

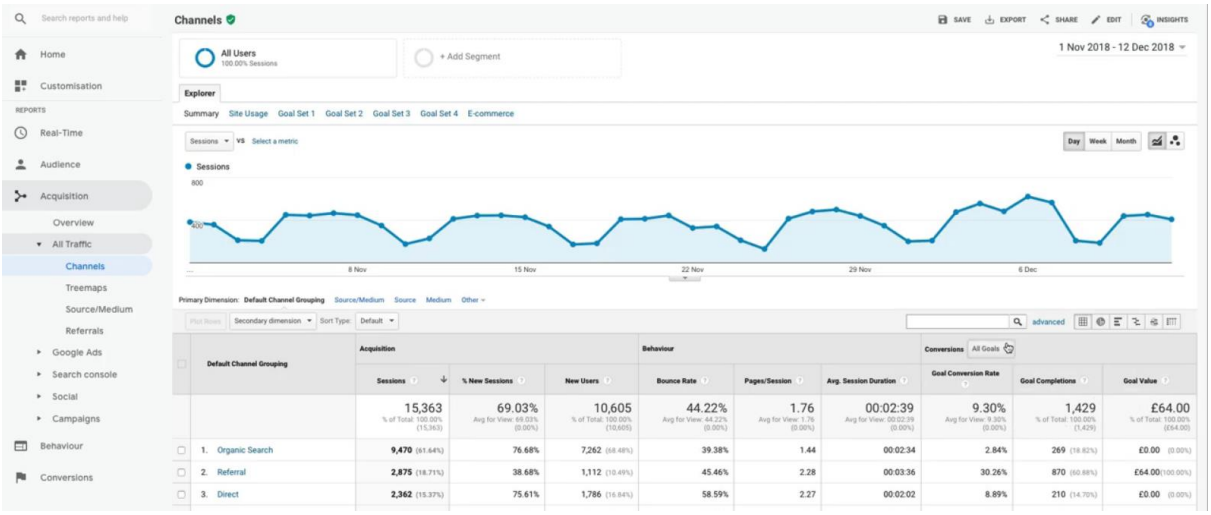


Рис. 2. Головне меню Google Analytics

Деякі з них будуть повторними відвідуваннями, деякі з них будуть абсолютно новими відвідуваннями, тому даний додаток надає змогу побачити відсоток повторних сесій.

Даний додаток також надає змогу дізнатися про кількість користувачів, що перейшли з веб-сайтів, що співпрацюють за реферальною програмою, а також із соціальних мереж.

В цілому Google Analytics на сьогодні є найбільшим додатком в своїй сфері. Його успішність полягає в тому, що велика кількість функціональних можливостей реалізована за допомогою експлуатації Google API а також найбільшого продукту компанії, що створила даний додаток – пошукової системи Google.

Перевагою даного додатку є те, що він надає різноманітну статистику користувацьких дій.

Недоліком даного додатку є те, що зібрана статистика лише подається в зручному для перегляду вигляді, проте не містить в собі жодних висновків. Через це додаток є доволі складним у використанні, і потребує певну кількість кадрових ресурсів для роботи з ним.

1.1.2. *Hotjar*

Даний додаток має іншу спрямованість своєї діяльності. Він орієнтований на візуальне відображення користувацьких дій на веб-сайті. Головною відмінністю даного додатку від інших є те, що вся інформація відображена у вигляді мап кліків. Велика кількість підприємців користуються даним додатком. На даний момент Hotjar знаходиться в стадії бета-розробки, але вже впевнено зарекомендував себе серед популярних комерційних веб-сервісів.

Основними метриками, які надає даний додаток, є:

- мапа кліків користувачів (рис. 3);
- мапа перебування курсору користувачів;
- мапа скролів користувачів (рис. 4).

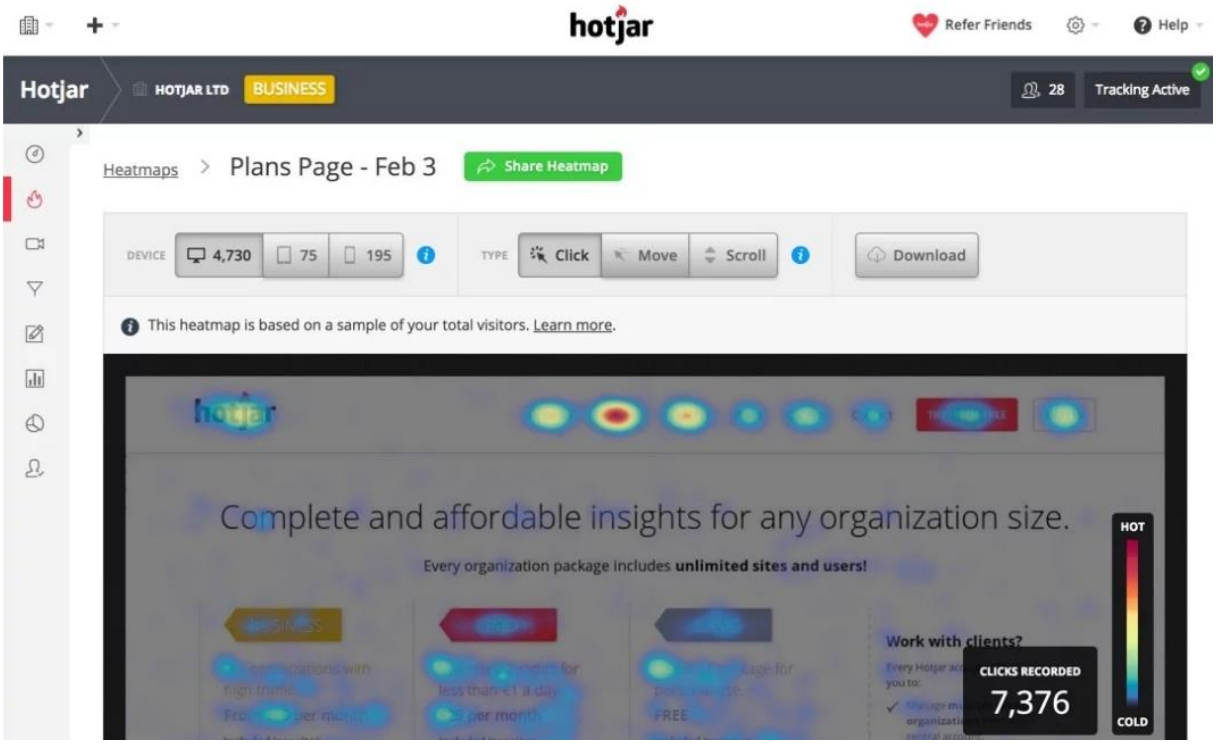


Рис. 3. Мапа кліків користувачів

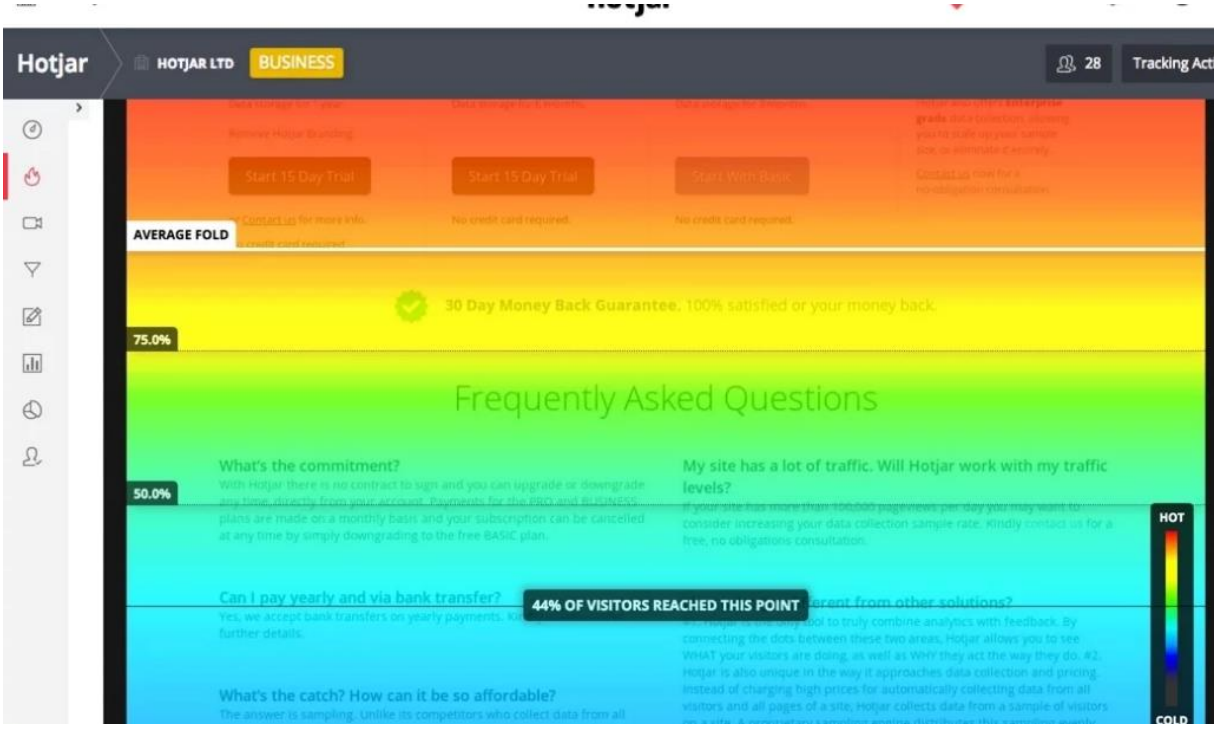


Рис. 4. Мапа скролів користувачів

Перевагою даного додатку є те, що він надає візуалізовану інформацію для користувача та власнику веб-сайту, на якому він

встановлений. Недоліком даного додатку є те, що він збирає неповну інформацію та не має можливостей розширяти параметри пошуку користувацьких дій в собі. Як і попередньо розглянутий додаток, даний додаток не пропонує жодних висновків по зібраній інформації, проте візуалізація зібраної статистики в кольоровому вигляді робить цей процес значно швидшим [1].

1.1.3. *AWstats*

Даний додаток є open-source застосунком, що означає, що він є безкоштовним і його програмний текст можна змінювати. Даний додаток є найпопулярніші вибором IT-фахівців у світі. Оскільки цей додаток є Open Source, окрім розробників, велику кількість функціональних можливостей вже дописали інші користувачі. Обсяг можливостей даного додатку постійно зростає. Даний додаток є найбільшим конкурентом Google Analytics оскільки надає повну статистичну інформацію користувацьких дій.

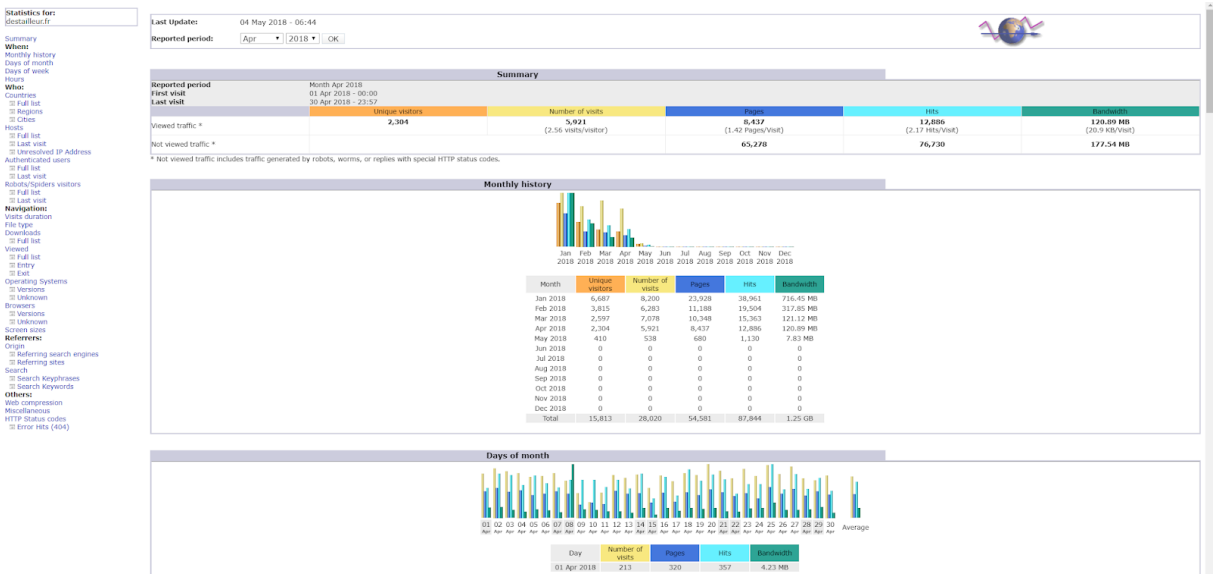


Рис. 5. Головна сторінка адміністраторського модуля AWStats

Основні метрики AWStats:

- кількість відвідувачів за обраний проміжок часу;
- кількість унікальних відвідувачів за обраний проміжок часу;
- кількість afk-сеансів;
- основні канали приходу користувачів на даний веб-сайт;
- drop rate по кожній сторінці веб-сайту;
- середній час сесії користувача на веб-сайті;
- кількість кліків за годину або інший обраний проміжок часу.

Перевагою даного додатку є те, що він безкоштовний і Open Source. Через це нові функціональні можливості створюються великою кількістю розробників у швидкому темпі.

Недоліком даного додатку є те, що не пропонується жодних висновків у зібранні користувацькі статистиці. Оскільки даний додаток є open-source, він має найкращий дизайн і подає інформацію не в читабельному вигляді.

1.1.4. *Додаток «SmartLook»*

Даний сервіс надає можливість привілейованим користувачам (адміністраторам, консультантам) відстежувати будь-яку дію користувача на веб-сайті.

SmartLook надає консультантам наступні можливості:

- бачити в режимі онлайн, що пише користувач в текстфілдах, розміщених на сайті;
- відстежувати будь-який рух курсору на сайті;
- автоматичне створення відео того, що робив користувач на сайті;
- створення мапи кліків на веб-сайті.

Серед недоліків слід зазначити:

- сервіс не збирає і не формує статистику по відвідуванням користувачів самостійно;

- відсутність алгоритмів, що відокремлюють зацікавленого користувача від звичайного відвідувача;
- сервіс не надає функцій чату.

Даний сервіс є джерелом статистики активності користувачів. Але через недостатню автоматизованість своєї роботи для успішною реалізації даного додатку потрібна велика кількість консультантів, які будуть відстежувати, чи спромігся користувач знайти потрібну інформацію або функцію веб-сервісу. За допомогою засобів AngularJS, React, Ember та використовуючи підхід AJAX, в режимі онлайн створюються записи користувацької активності. Веб-сторніки користувачів зберігаються у віддаленому режимі [2].

Особливістю даного додатку є наступне:

- можливість запису відео-активності користувача;
- можливість підтримки до 100 тис. запитів на секунду;
- даний застосунок є веб-сервісом, отже, операційна система, якою користується консультант не має впливу на його роботу.

SmartLook є вибором для великих комерційних сервісів, оскільки здебільшого його головною перевагою є можливість підтримки потужних навантажень на систему, що стає можливою завдяки використання PostgreSQL баз даних. Даний сервіс встановлюється на більшість сучасних CMS систем, що є PHP-based.

1.1.5. *Додаток «OpenWebAnalytics»*

Даний застосунок використовується власниками веб-сервісів для створення користувацької статистики.

Даний сервіс можна використовувати на веб-сервісах WordPress, MediaWiki. За допомогою засобів PHP 5.2.x та старших версій даний застосунок надає можливість відстежувати активність користувачів при перегляді різних WordPress веб-сайтів.

Даний застосунок реалізований на двох мовах: PHP, JavaScript. Його головний модуль OWA.tracker.js містить в собі всю функціональність даного сервісу, а робочий клас OWA – набір інструментів зі сторінкою користувача.

Недоліком даного модуля є те, що він не підтримує методологію AJAX. Методи, що забезпечують основну роботу даного додатку:

- OWA.tracker() – ініціалізація класу, що відповідає за всі функціональні можливості;
- OWATracker.trackPageView() – метод, що відстежує будь-яку дію користувача на сторінці, та будь-яка дія користувача на сайті автоматично надішле повідомлення на сервер. Серед можливих дій існуються такі, як вихід користувача з веб-сайту, спроба авторизуватися, перехід на інші сторінки веб-сайту, створення коментарів, тощо.
- OWA.trackDomStream() – метод, що починає відстеження кожної дії користувача, яка була зроблена за допомогою миші. Недоліком його є те, що інформація, отримана за допомогою цього метода, явним чином не використовується в додатку.

В OpenWebAnalytics використовується MySQL 4.1+ бази даних. База даних створена за типом зірки (star schema database). Розглянутий додаток має відкритий сирцевий код та не вимагає ліцензії на користування. Клієнтами OpenWebAnalytics є молоді комерційні сервіси.

1.2. Висновки

В даному розділі було розглянуто основні аналоги веб-сервісів, що надають статистику по користувацьким діям на веб-сайті на якому були встановлені. Основним недоліком більшості з них є те, що такі додатки потрібно створювати окремо під кожну платформу. Окрему увагу слід звернути на додаток Google Analytics, який в своїх функціональних

можливостях використовує створені компанією, що його розробляла, продукти. Перевагою переглянутих додатків є великий обсяг статистичної інформації, яку можна відстежувати, а також зручний графічний вигляд, в якому вона подається.

Основним недоліком у всіх вище переглянутих додатків є те, що жоден з них не пропонує аналізу та висновків зібраної статистики. В ході ведення бізнесу це спричиняє те, що власник веб-сайту повинен наймати окремих працівників, що будуть аналізувати статистику, яку надають дані додатки. Робота працівників є несистематизованою та неорганізованою, що означає, що кошти, що витрачаються на таких працівників, є малоефективними.

2. ФОРМУЛЮВАННЯ ПРОГРАМНОГО МЕТОДУ ДЛЯ ОПТИМІЗАЦІЇ ТА ФОРМАЛІЗАЦІЇ ПОБУДОВИ ІНТРЕФЕЙСІВ КОРИСТУВАЧА ДЛЯ КОМЕРЦІЙНИХ ВЕБ-ДОДАТКІВ

2.1. Огляд підходів до побудови інтерфейсів веб-сайтів

Інтерфейс користувача (ІК) – це сукупність засобів, за допомогою яких користувач взаємодіє з різними пристроями (з комп’ютером або побутовою технікою) або іншим складним інструментарієм (системою). Інтерфейс користувача – це такий різновид інтерфейсів, в якому з одного боку – людина, з іншого – машина (пристрій, програмне забезпечення). За визначенням Національного банку стандартизованих науково-технічних термінів, інтерфейс користувача – це комплекс апаратних і програмних засобів, що забезпечує взаємодію користувача з комп'ютером.

ІК часто розуміють лише як зовнішній вигляд програмного заебзпечення (ПЗ), але таке розуміння є надто вузьким, оскільки саме за допомогою інтерфейсу користувач сприймає програму в цілому та використовує її функціональність. ІК забезпечує підтримку прийняття рішень у визначеній предметній галузі та визначає порядок використання ПЗ і документації до нього. В дійсності, ІК об'єднує усі елементи і компоненти ПЗ, які здатні впливати на взаємодію користувача з програмним забезпеченням. До таких елементів належать: набір задач, які користувач розв’язує за допомогою ПЗ; використовувана програмним забезпеченням метафора (наприклад, "робочий стіл" у операційній системі Windows); елементи управління ПЗ; навігація між блоками ПЗ; візуальний (і не тільки) дизайн вікон та екранних форм програми та інші складові.

Стиль інтерфейсу користувача – це набір ознак, методів, прийомів діяльності, які характеризують індивідуальність інтерфейсу користувача, а також сукупність прийомів використання інструментів розроблення ПЗ.

Процес проектування ІК – це складний, нелінійний, недетермінований процес. Складність ІК обумовлюється рядом

невизначеностей, які суттєво впливають на процес розроблення програмного забезпечення. Нелінійність проектування ІК полягає у відсутності фіксованого, впорядкованого і прямолінійного алгоритму від початку до кінця проектування. Процес проектування є невизначеним, оскільки не існує рівняння, за яким можна було б одержати однаковий результат при заданих однакових початкових умовах, більш того, одержати ідентичний результат практично неможливо. Інтерфейс користувача неортогональний у тому сенсі, що будь-який аспект проектного рішення може впливати на інші аспекти, до того ж результат цього впливу не завжди є позитивним та прийнятним.

Процес проектування сучасного ПЗ передбачає вирішення ряду задач, зокрема: зниження витрат на проектування, скорочення термінів проектування, покращення якості пропонованих рішень, забезпечення нескладного в освоєнні та використанні ПЗ, вивчення та впровадження нових технологій та засобів, досягнення кращих результатів в порівнянні з конкурентами.

Супроводження ІК – це процес покращення, оптимізації та усунення недоліків ІК після передачі програмного забезпечення в експлуатацію. Пристосованість ІК до супроводження важлива можливістю покращення ІК вже за участі користувачів. До інших факторів відносяться узгодженість, інтегрованість та вартість ІК, які впливають на задоволеність користувача інтерфейсом, а відтак і програмним продуктом в цілому.

Усі фактори задоволеності користувача та їх відносну важливість слід враховувати під час кожного етапу життєвого циклу програмного забезпечення ІК.

До труднощів проектування ІК слід віднести і той факт, що користувач не завжди може чітко висловити свої вимоги та побажання щодо програмного продукту та його інтерфейсу на етапі проектування, але є дуже категоричним щодо бажаності і небажаності тих чи інших властивостей на етапі введення ПЗ в експлуатацію.

Часто характеристики ІК щодо практичності, інтеграції та узгодженості не формулюються явно на етапі проектування ПЗ, а визначаються на рівні деяких очікувань, що призводить до невірного розуміння очікувань замовника проектувальниками ПЗ. Тому такі вимоги повинні визначатись явно, причому бути вимірюваними (мати кількісні характеристики), оскільки проектувальник ІК може не бачити і не розуміти видимі та зрозумілі замовникам і користувачам вимоги.

В минулому розроблення ІК розвивалось лише шляхом еволюції технологій та систем, на базі яких розроблялось ПЗ. Такий підхід називають системно-керованим або технологічно-керованим. Побажання користувачів абсолютно не враховувались, їм надавались програмні функції з інтерфейсом, який розробники були в стані розробити.

Однією з головних причин створення невдалого програмного забезпечення є недостатнє залучення користувачів до проекту. Не менше значення мають і наслідки недостатньо активної участі користувачів у проекті, зокрема відсутність знань про фактичних користувачів продукту та середовище використання розробленого ПЗ. Детальна інформація про користувачів та їх середовище допомагає встановити рамки, в яких повинно здійснюватись проектування ІК та забезпечуватись його практичність. Поряд з вимогами до ІК та його практичності інформація про користувачів та їх середовище допомагає колективу розробників виділити ті особливості продукту, яких потребують користувачі, що є важливим для вибору відповідних методів розроблення ІК та підходів до проектування спільного стилю додатків. З початку 80-х років при розробленні ПЗ акцент було перенесено на користувача, причому користувачі залучались до розроблення. Однак їм відводилась пасивна роль: у них з'ясовували, які вимоги вони висувають до ПЗ і які задачі вони вирішуватимуть за його допомогою. Зараз більшість розробників дотримуються методології, котру називають "розробленням із залученням користувачів" та "розробленням, орієнтованим на користувачів, що

навчаються". Новизна підходу полягає в тому, що користувачів розглядають як активних учасників самого процесу розроблення. Залучення користувачів сприяє підвищенню доступності інтерфейсу та програмного засобу, а також служить гарантією, що одержане ПЗ буде відповідати запитам та вимогам.

Розроблення, орієнтоване на користувачів, що навчаються, спрямоване на те, щоб в процесі вирішення своїх задач людина навчалась новим навичкам роботи з ПЗ, тобто на її інтелектуальний розвиток, тренування її уяви і одержання знань у різних галузях.

Розроблення, орієнтоване на користувача, базується на наступних керівних принципах:

- розуміння потреб користувачів є рушійною силою усього проекту;
- все, що користувачі бачать і до чого вони мають доступ, повинно проектуватись сумісними зусиллями;
- інноваційний проект завжди є результатом інтенсивної роботи команди спеціалістів з різних галузей;
- рішення щодо ІК повинні базуватись на зворотньому зв'язку з користувачами;
- результати зворотнього зв'язку з користувачами повинні збиратись з заданою точністю, швидкістю та частотою;
- зворотній зв'язок здійснюється як з потенційними, так і з фактичними користувачами;
- розроблення, орієнтоване на типового користувача, повинне стандартизуватись і впроваджуватись;
- розроблення, орієнтоване на користувача, повинне постійно вдосконалюватись.

Базові принципи проектування, орієнтованого на користувача:

- розуміння користувачів та їх задач, залучення користувачів на різних етапах життєвого циклу ПЗ;
- визначення цілей, які можна виміряти; встановлення критеріїв успіху з точки зору користувачів та замовників;
- проект повинен передбачати нову компетентність користувача, яка по відношенню до продукту включає пакетування, маркетинг, навчання, надруковану інформацію, налагодження параметрів, інсталяцію, екранні форми (екранна форма – графічно відображена форма (локація) в програмному забезпеченні; цифрове зображення, одержане програмним забезпеченням за командою користувача), графіку, довідки, іншу експлуатційну підтримку, оновлення та деінсталяцію;
- оцінювання та тестування ПЗ за участю фактичних користувачів для визначення, чи досягнуті цілі та які є проблеми;
- ітеративний підхід – якщо цілі не досягнуті або існують проблеми, слід внести виправлення та провести повторну перевірку.

Будь-який процес розроблення ІК повинен бути ітераційним, оскільки вдалий інтерфейс неможливо одержати без періодичного повернення до попередніх етапів. Критерієм для завершення ітераційного розроблення є той факт, що усі вимоги користувачів задоволені, а сам продукт відповідає запланованим цілям. Може здаватись, що ітераційний процес займає багато часу через багаторазові проходження етапів розробки. Однак початкові проходження етапів допомагають створити варіанти розробок та прототипів, які в наступних ітераціях зекономлять час на впровадження та тестування [3].

Усі методи оцінювання пов’язані із залученням потенційних користувачів програмного продукту.

Загальні критерії досягнення цілей створення ІК повинні бути чітко визначені, зрозумілі й прийняті замовниками та розробниками; досягнення поставлених цілей може вимагати багатократних ітерацій впровадження та підтримки.

На етапі впровадження здійснюється оцінка ПЗ за участю користувачів, які не залучались до розроблення, пілотне тестування, виконання задач, які не оцінювались або не були передбачені під час проектування та розроблення.

Для створення стилів ІК та успішного створення ПЗ потрібен висококваліфікований персонал, який володіє широким набором різноманітних спеціальних навичок в таких галузях: проектування ІК; технологія розроблення ПЗ та ІК; тестування та оцінювання якості ПЗ та ІК; стандартизація ПЗ та ІК; інструментальні засоби реалізації та використання ІК в додатках; проектування та реалізація візуальних та графічних конструкцій; проектування та реалізація засобів навчання, керівництв, систем допомоги та електронної експлуатаційної підтримки; психологія та ергономіка, людська поведінка, сприйняття, навчання та пізнання; бізнес-планування; управління проектами. Отже, для успішного проектування ІК потрібен колектив розробників, кожен з яких є професіоналом в певній галузі і володіє наступними якостями: вміння працювати в команді, причому працювати як з розробниками, так і з комерційними структурами; вміння розуміти користувачів, уточнювати і деталізувати нечітко сформульовані вимоги до ІК; навички визначення і погодження з користувачем кількісної оцінки практичності, інтегрованості та узгодженості ІК; вміння використовувати засоби і методи розроблення ПЗ; навички використання засобів і методів тестування ПЗ інтерфейсу користувача, розрахунку надійності та якості ПЗ інтерфейсу користувача; навички швидко і ефективно оцінювати і відновлювати проект та реалізацію [4].

Проблеми організаційного, групового та індивідуального характеру, які виникають при проектуванні ІК, відрізняються набагато більшою складністю, ніж технічні проблеми. 80% проблем пов'язані з людьми і лише 20% носять технічний характер [5].

Значення різних професійних навичок та участь у проекті відповідної кількості спеціалістів, які володіють кожним типом навичок, потребує ретельного планування для досягнення успішності проекту. Правильне визначення ролі розробника в колективі розробників дозволяє усунути невизначеності. Один спеціаліст може виконувати одну або декілька ролей.

Стиль взаємодії “користувач-комп’ютер”, в якому застосовуються 4 елементи: вікна, піктограми, меню та покажчики, які разом складають графічний інтерфейс користувача (ГІК). Іноді ГІК називають WIMP-інтерфейсом (Windows, Icons, Menus, Pointers). Найважливішими властивостями ГІК є можливість безпосереднього маніпулювання об'єктами, підтримка миші та інших вказівних пристроїв, використання графіки та наявність області для функцій і даних ПЗ. Кожному ПЗ властиві свої унікальні принципи побудови ГІК, причому всі додатки повинні бути витримані в цьому стилі. ГІК не гарантує більш високого рівня практичності. Належним чином спроектований ГІК-орієнтований додаток може перевершувати аналог з неграфічним інтерфейсом за рахунок підвищення ефективності роботи користувача та ступеня його задоволеності.

Веб-додатки мають всі влстивості ГІК, їх інтерфейс називають Web User Interface (WUI). Навігація виконується в рамках одного або декількох додатків з використанням текстових або візуальних гіперпосилань. Основні особливості WUI-додатків: інформація відображається в єдиному вікні (браузері), браузер забезпечує меню для Web-додатку; вибір дій обмежений; веб-сторінка має невисокий ступінь внутрішнього контролю над клієнтською областю для відкриття спеціалізованих контекстних

меню; створення спеціалізованих меню потребує додаткового програмування; функціональні можливості додатку повинні відображатись в методи для виклику команд; клієнтська область не містить традиційних піктограм; деякі додатки використовують графіку та анімацію в естетичних або навігаційних цілях, що несе в собі загрозу зовнішнього візуального шуму і збільшення часу відгуку при завантаженні та відкритті файлів; браузер і додатки забезпечують можливість відключення графіки; технологія drag-and-drop не підтримується, обмежені дії правої кнопки миші. До найбільш поширених компонентів WUI-інтерфейсів відносяться банери (заголовки), навігаційні панелі та візуальні або текстові гіперпосилання.

Побудований на основі ІК програмний додаток в тій чи іншій мірі використовує компоненти стилю ІК або компоненти ІК прикладного рівня. Заснований на застосуванні ІК додаток здійснює виведення інформації та забезпечує взаємодію з користувачем за допомогою екрану та надає користувачеві піктограми та курсор. Прикладний рівень додатку, заснованого на застосуванні ІК, містить всі функціональні можливості, які виходять за межі оформлення діалогових вікон та реалізовані із застосуванням різних стилів ІК. Можливості прикладного рівня включають наступні аспекти: концептуальне проектування; семантику об’єктів і операцій; наочне представлення та стиль поведінки об’єктів в межах клієнтської області; синтаксис об’єктів – формати даних, діапазони вхідних даних та послідовність зміни вікон; методи взаємодії та безпосереднього маніпулювання; унікальні дії; використання фізичних пристроїв.

Проектування ПЗ дає можливість надати користувачу інтерфейс, який має об’єктно-орієнтований стиль та/або об’єктно-орієнтовану внутрішню структуру (реалізацію). Більшість об’єктно-орієнтованих властивостей знаходять відображення в зовнішньому вигляді, поведінці, вимогах до взаємодії та функціональних можливостях ІК. Об’єктно-

орієнтований прикладний ІК повинен мати наступні властивості: забезпечувати пряме маніпулювання, забезпечувати пряме введення даних, забезпечувати контекстну залежність інтерфейсу від об’єктів.

Використання програмних продуктів вимагає мінімальних зусиль для вирішення задач користувача. ІК реалізується таким чином, щоб уникнути складної ієрархічності вікон та/або екранів, а також зайвих дій з клавіатурою та мишею.

Програмне забезпечення повинно бути доступним в різних формах для задоволення індивідуальних потреб. Програмні об’єкти ІК повинні мати можливість налагодження (налагоджуваність). Якісний прикладний ІК дозволяє користувачу обрати метод взаємодії і методи макетування та доступу для оптимізації потреб користувачів.

Для опису макету потреб користувачів часто використовують ментальну модель. Ментальна, або концептуальна модель відповідає на питання "як користувач розуміє і взаємодіє з ПЗ". ІВМ константує: "Ментальна модель необов'язково точно відображає ситуацію та її компоненти на даний момент часу. Вона допомагає передбачити, що відбудеться далі, служить основою для розуміння, аналізу і прийняття рішень".

Ментальні моделі дозволяють користувачам передбачити (або позначити невидимі) події; знайти причини позначених подій; визначити необхідні дії для здійснення потрібних змін; використовувати їх як мнемонічні пристрої для запам’ятовування подій та зв’язків (відношень); забезпечити розуміння аналогічних пристроїв. В багатьох відношеннях концептуальна модель ІК – це фактично його архітектура. Послідовні рівні проектних рішень дозволяють нарощувати ступінь деталізації зовіншнього вигляду, поведінки, інформаційної підтримки та взаємодії користувачів на більш пізніх етапах проектування. Концептуальне проектування, яке стосується концептуальних моделей, стилю ІК та навігаційних структур, не обмежується деталями зовнішнього вигляду, поведінки, взаємодії і на

функціональних можливостях продукту. Перший вірний крок важливий для збільшення шансів на досягнення цілей проекту у відношенні ІК, практичності, узгодженості, інформаційної підтримки, часу відгуку, надійності та інших проектних факторів.

Альтернативні концептуальні рішення формулюються, оцінюються і піддаються ітеративному перегляду, а також швидкому нарощенню при невеликих витратах – на концептуальному рівні. Однак помилка в концептуальних проектних рішеннях, не виявлена на ранніх проектних етапах розробки, передається на наступний етап і потенційно впливає на множину екранних форм, взаємодій, підтримку користувачів, програмний код, навчання, тестування, план-графік та вартість розробки. Вартість виправлення такої помилки буде тим вищою, чим довше помилка залишається невиявленою.

Концептуальна модель являє собою попередній ескіз, який показує основні функції ІК, візуальні, інформаційні та інші характеристики проекту продукту. Архітектура ІК – це розподіл функцій та програмний інтерфейс між ПЗ та користувачем.

На основі аналізу характеристик користувачів, вимог, задач та ділових потреб можна представити в явній і наочній формі основну ідею проекту, організаційні принципи та структуру, а також типові поведінкові моделі ПЗ по відношенню до візуалізації та взаємодії.

Склад концептуальної моделі представлений на рис. 6.

Основна задача, пов'язана з концептуальним проектуванням, полягає в тому, щоб встановити, які функції покладаються на користувача, а які – на ПЗ.

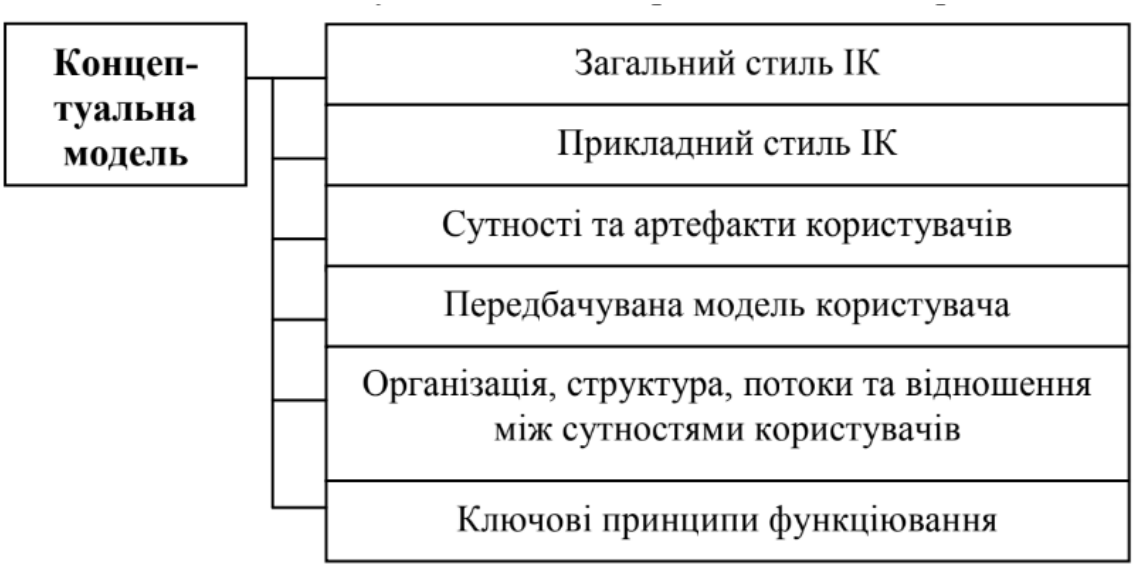


Рис. 6. Структура концептуальної моделі

Результат цієї задачі визначається наступними аспектами:

- задачі, які підлягають виконанню;
- користувачі, які виконують задачі;
- сильні сторони та обмеження можливостей користувачів;
- сильні сторони та обмеження апаратного забезпечення, операційної системи, ПЗ, засобів ІК та стилів взаємодії, які належать задачі;
- перспективні характеристики ПЗ, які покращують, нарощують або автоматизують окремі аспекти розв'язку задач користувачів;
- перспективний шлях, який веде до досягнення заданих часових та вартісних обмежень;
- перспективи виконання вимог по відношенню до ІК та практичності.

Модель проектувальника – це дещо середнє між моделлю користувача та моделлю програміста. Проектувальник дізнається про ідеї, побажання користувача, поєднує їх зі своїми навичками і матеріалами та проектує ПЗ, яке повинно задовольняти потреби користувача. Модель

проектувальника описує об’єкти користувача і техніку маніпулювання ними.

Традиційними методами залучення користувачів до розроблення ПЗ є їх участь у проекті на етапах аналізу вимог та його оцінки. Однак існує ряд простих і ефективних методів залучення користувачів на більш ранніх етапах, а то й протягом усього життєвого циклу розроблення ПЗ.

На етапі планування користувачі можуть допомогти у вирішенні проблем: складання уявлення про кваліфікацію і навички співробітників та керівників; потреби у додатковому залученні користувачів до проекту; вказання додаткових джерел вхідної інформації; вказання вимог користувачів, даних і процедур; точка зору користувачів на пріоритети та вимоги до ПЗ.

На етапі аналізу вимог користувачі допомагають у визначенні профілів користувача, ділового контексту, потоків задач, стереотипів робочої поведінки; виявленні найбільш часто виконуваних задач і функцій; виявленні проблем з використовуваними платформами та операційними системами; вдосконаленні та зміні бізнес-процесу; формуванні послідовності екранних форм та кроків діалогів; розташуванні функціональних кнопок та даних на екрані; виявленні переваг по відношенню до графічних представлень та термінології та впливів при керуванні змінами. На етапі проектування користувачі можуть приймати участь у розробленні таких проектних рішень, як поведінка, інтеграція і узгодженість робочого середовища ІК; концептуальні моделі, термінологія і піктограми; послідовності екранних форм – кроки/вміст, графічний стиль, компонування; всі деталі, що стосуються ІК, а також загальний досвід користувачів.

На етапі реалізації користувачі допомагають при вирішенні наступних задач: чіткість представлення графічних елементів ІК на пристроях відображення; кількість елементів в списках вибору; кількість елементів у списку, який повертає операція пошуку; початковий час

відгуку при виклику зображення; непередбачувана поведінка програмного забезпечення; незвичайні формати друку.

Розглянемо вплив людського фактору. Під людським фактором розуміється вивчення людини та використання одержаної інформації для проектування засобів, задач і середовища ПЗ з метою забезпечення продуктивної та комфортної роботи [5].

Проектування ІК пов'язане з діяльністю двох категорій людей – розробників та користувачів. Розробники ІК, в свою чергу, теж є користувачами іншого прикладного ПЗ. Отже, існують можливості вдосконалити ІК як для розробників, так і для користувачів. Розробники можуть використовувати більш досконалі методи і засоби проектування програмного забезпечення ІК, а користувачі матимуть можливості одержати всі переваги готового ПЗ.

Характеристики комп’ютерних засобів введення, обробки і виведення інформації істотно відрізняються, але ще більші відмінності існують в характеристиках введення, обробки і виведення інформації користувачами. Розробники повинні розуміти, що користувачі мають як сильні сторони, так і обмеження і слабкості, пов'язані з віком, станом здоров'я, пам'яттю та іншими факторами. Отже, слід проектувати ІК, враховуючи слабкі сторони користувачів, лише тоді можна одержати ефективне ПЗ.

Недостатність уваги до людського фактору призводить до створення непродуктивного, важкого у вивченні та у використанні ПЗ.

Ергономіка вивчає взаємозв’язки між людиною та її роботою [5]. Цей термін використовується як синонім дослідження людського фактора.

Зручність роботи не надає такої користі бізнесу, як продуктивність роботи, але продуктивність підвищуватиметься, якщо користувачу буде зручно працювати. Для вимірювання продуктивності роботи користувача використовуються метрики (наприклад, час, необхідний для виконання задачі без помилок). Вимірювання комфортності можливе лише в термінах

рівня задоволеності користувача та його думок з приводу практичного використання ПЗ.

В контексті проектування ІК мета застосування ергономічних принципів полягає в підвищенні ефективності та задоволеності користувача інтерфейсом. Фактори, які належать до апаратного забезпечення, сконцентровані на фізичних та антропометричних характеристиках та особливостях людей. При проектуванні ПЗ слід брати до уваги ряд притаманних людині властивостей нефізичного характеру та враховувати психологічні характеристики людини. Ергономіка та людські фактори ПЗ призначені для опису результатів вивчення інформації про користувача та застосування цієї інформації при проектуванні ПЗ з метою підвищення ефективності та задоволеності користувачів.

Існує два основних принципи ергономіки щодо ПЗ:

- навчання і набуття практики людиною з метою адаптації до існуючого ПЗ;
- проектування ПЗ, яке відповідає можливостям його користувачів.

Для проектування інтерфейсу, орієнтованого на користувача, використовується другий принцип ергономіки, тому що саме інтерфейс повинен налагоджуватись на потреби користувачів.

Незалежно від того, є користувач розробником ПЗ чи він працює з прикладним ПЗ, необхідно детально дослідити особливості візуального сприйняття та опрацювання ним інформації [6].

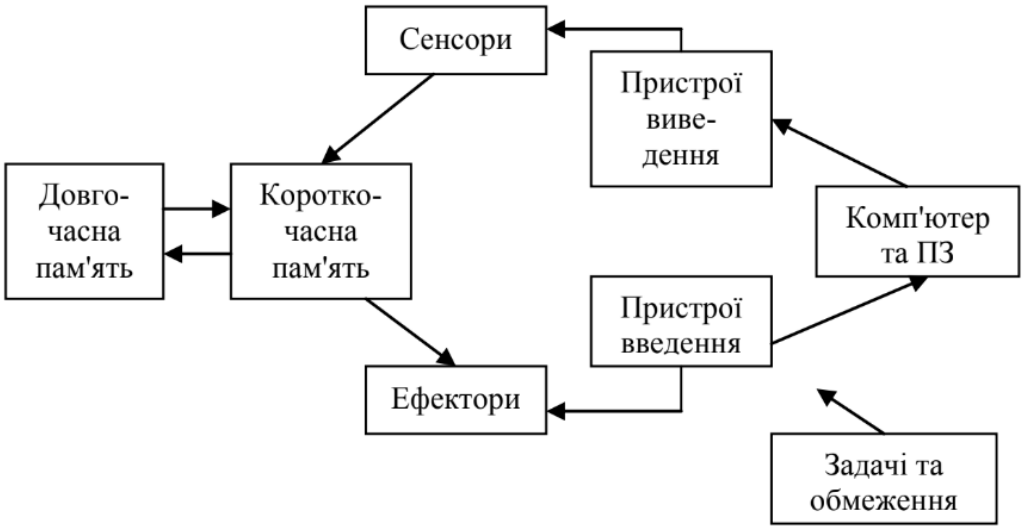


Рис. 7. Модель опрацювання інформації користувачем ПК

Провідний інженер з розроблення програмного забезпечення та інтерфейсів користувача компанії Cisco Systems Трейсі Леонард сказав: “Золоте правило проектувальника: “Ніколи не робити іншим того, що не подобається тобі". Згадайте, що вам не подобається в ПЗ, яким ви користуєтесь, і не робіть того ж в програмі, над якою працюєте”.

Однією з актуальних вимог до сучасного ПЗ є необхідність налаштування інтерфейсів до вимог та потреб користувачів. Кожен позитивний досвід спілкування користувача з ПЗ дозволяє йому підвищувати рівень компетентності та навички.

Основними принципами розробки інтерфейсу користувача є:

- контроль інтерфейсу користувачем;
- зменшення навантаження на пам’ять користувача;
- послідовність інтерфейсу.

Хансен подає наступний список принципів проектування інтерфейсу:

- скорочення об'єму інформації для запам’ятовування;
- оптимізація операцій;
- мінімізація помилок.

Ці принципи застосовні до усього програмного забезпечення, в усіх типах та стилях інтерфейсів. Трактування цих принципів залежить від операційної системи, складових інтерфейсу користувача та його задач. Моделі користувача теж різні і впливають на те, яким чином будуть застосовуватись вказані принципи. На деяких етапах розроблення проекту може постати питання: ”Що відбудеться далі?”. Відповідь повинна бути: “Те, чого захоче користувач”.

Розглянемо основні правила проектування інтерфейсу користувача. Основне правило проектування ІК – дати користувачу контроль над системою. Проста аналогія – водій автомобіля та пасажир потягу. Рішення їхати власним авто або потягом – повинен приймати саме пасажир (в нашому випадку користувач). Розробник інтерфейсу повинен виділити в якості основного такий принцип проектування, при якому продукт буде задовільняти всім вимогам. Досвідчені проектувальники дозволяють користувачам вирішувати деякі задачі на власний розсуд. Так, наприклад, європейські архітектори по завершенню будівництва складного комплексу будівель повинні прокласти між ними доріжки для пішоходів. На майданчиках між будинками ставлять таблички: “Ходіть, будь-ласка, по траві”. Через деякий час будівельники повертаються і лише тоді, згідно волевиявленню населення, асфальтують протоптані доріжки. Це гарний приклад надання можливості користувачу контролю над ситуацією і проектування інтерфейсу, який би відповідав потребам та побажанням користувача.

Основними положеннями надання контролю користувачу над інтерфейсом є безрежимність, гнучкість, можливість переривання, корисність, поблажливість, можливість орієнтування, доступність, спрощення користування, адаптованість, інтерактивність.

Безрежимність – вибір та використання режимів розважливо. Режими (певні умови роботи, діяльності) – атрибут багатьох програмних інтерфейсів, але застосовувати їх потрібно лише при необхідності.

Існує ряд інтерфейсів, які занадто часто перемикаються між режимами. На екрані з’являється діалогове вікно режиму, і користувач стає обмеженим в діях не лише в даній програмі, але й не має можливості переходу в інші програми [7].

Інтерфейс може працювати в двох режимах, які в багатьох випадках необхідні, однак позбавляють користувача самостійності:

- режим користувача – обмежує режим роботи і дозволяє користувачу виконувати лише певні дії в певному місці програми. В наш час проектувальники інтерфейсів все частіше обходяться без перемикання режимів. Можливо, оптимальним варіантом є показ даних в форматі, який відповідає рівню доступу користувача;
- системний режим – використовується достатньо рідко. Якщо система знаходиться в цьому режимі, дозволено працювати лише з поточною програмою. При розробленні повідомлень та інформації про допомогу в такому режимі слід пам’ятати про те, наскільки незручний системний режим для користувача.

При виборі режимів важливо слідувати принципу негайного візуального оберненого зв’язку. Користувач повинен бути постійно впевнений в тому, що він знаходиться в потрібному режимі. Режими інтерфейсу повинні бути настільки природніми, щоб користувачу було комфортно працювати з ними.

Гнучкість – забезпечує користувачу можливість вибору. Наприклад, можливість роботи з клавіатурою передбачає використання клавіатури замість миші. Панелі інструментів створені для прискорення роботи при використанні миші, однак при роботі з клавіатурою важко дістатись – для подібних випадків передбачені підменю. Більшість користувачів мають звичку працювати як з клавіатурою, так і з мишею.

Можливість переривання – дозвіл користувачу перемикати увагу між різними програмними засобами. Програмні інтерфейси повинні бути спроектовані так, щоб користувач міг в будь-яку хвилину перерватись або зберегти результати роботи. Забезпечення контролю користувача над програмою та його підтримка – ось головні принципи розроблення.

Не потрібно змушувати користувачів закінчувати виконання початих послідовностей дій. Вони повинні мати вибір – анулювати або зберегти дані і повернутись туди, де вони перервались.

Корисність – демонстрація повідомлень, які допомагатимуть користувачу в роботі. У інтерфейсі потрібно використовувати зрозумілі для користувача терміни. Даний принцип застосовний не лише для повідомлень, але й для усіх текстів на екрані: запрошень, інструкцій, заголовків, написів на кнопках. Усі текстові аспекти інтерфейсу повинні розроблятися спеціалістами, котрі мають знання з орфографії та лексики. При написанні системної та програмної документації, а також повідомлень слід обрати вірний тон. Невдалі термінологія та тон призводять до того, що користувачі звинувачуватимуть себе у помилках, що виникають при використанні ПЗ.

Поблажливість – створення умов для негайних та зворотніх дій, а також зворотнього зв'язку. Недостатність зворотнього зв’язку в більшості програмних продуктів змушує користувача витрачати багато сил на виконання поставленої задачі. Наприклад, у інтерфейсі командного рядка в MS DOS файли видаляються командою DEL. Виконання цієї команди потрібно перевіряти командою DIR, тобто зворотнього зв’язку з командою DEL користувач не має [8].

Кожен програмний продукт повинен мати функції відхилення (скасування) та повторення дії. Необхідно інформувати користувача, якщо дана дія не може бути відхилена, і, по можливості, дозволити йому альтернативну дію.

Можливість орієнтування – забезпечення доступу користувачу до будь-якої частини інтерфейсу. Користувач повинен мати можливість вільно орієнтуватись в інтерфейсі, мати доступ до будь-якої його частини, можливість пересуватись вперед і назад по низхідній та висхідній структурі інтерфейсу, отримувати зручні контекстні підказки там, де вони потрібні. Наприклад, панель задач Windows показує, які програми виконуються, і надає користувачу доступ до них.

Панелі інструментів, меню, палітри та інші елементи інтерфейсу операційних систем, набори програмних продуктів та різні утиліти – все це розроблено, щоб допомогти користувачам орієнтуватись в операційній системі та в просторі жорсткого диску. Системні та програмні “помічники” та “асистенти” теж пропонують допомогу для орієнтації в програмних функціях або задачах. Доступність – налагодження системи на користувачів з різним рівнем підготовки. Деякі програми пропонують спеціальні інтерфейси, які допомагають обрати рівень складності взаємодії. Наприклад, панель задач та підменю програм можуть бути стандартними або деталізованими в залежності від вибору користувача та типу виконуваної задачі. Для досвідчених користувачів потрібно передбачити швидкий доступ до функцій ПЗ.

Спрощення користування – створення зрозумілого, "дружнього" ІК. ІК – "міфічна" частина програмного продукту. При вдалому проектуванні користувачі не помічають інтерфейсу. Якщо інтерфейс розроблений невдало, користувачам доведеться докласти чимало зусиль для ефективного використання програмного продукту. "Прозорість" інтерфейсу забезпечується тим, що користувачу надається можливість користуватись об’єктами, відмінними від системних команд.

Адаптованість – надання можливості користувачу налагоджувати інтерфейс за своїм смаком. Користувач повинен мати можливість налагодження: способу представлення інформації на свій вибір (кольори, шрифти, розташування елементів), поведінки інтерфейсу (дії за

замовчуванням, макроси, кнопки) та інтерфейсних функцій (натискання кнопок чи клавіш, сполучення клавіш для швидкого вибору команд, мнемоніка, розташування кнопок миші для виконання команд).

Інтерактивність – дозвіл користувачу маніпулювати об'єктами інтерфейсу. Всюди, де це можливо, потрібно дозволяти користувачу безпосередньо взаємодіяти з об’єктами на екрані в природньому, натуральному стилі. Користувачі повинні комфортно почуватись при виконанні операцій та знати про передбачуваний результат.

Положеннями зменшення навантаження на пам’ять користувача є запам'ятовування, розпізнавання, інформування, швидкість, інтуїтивність, перенос, терпимість, контекст, організація.

Запам’ятовування – не слід завантажувати короткочасну пам'ять. Короткочасна пам’ять допомагає зберігати інформацію протягом невеликого проміжку часу. Користувачі часто працюють над декількома задачами одночасно, тому не варто ІК завантажувати короткочасну пам’ять користувача в моменти перемикань між ними. Цей принцип проектування часто порушується, що змушує застосовувати зовнішні “засоби” для зберігання інформації (папір, калькулятор).

Функції програм (виділення останньої дії та її повтор) та дії з використанням буферу обміну дозволяють користувачам маніпулювати частинами інформації, необхідними в багатьох місцях, а також всередині задач. Оптимальним є варіант, коли програма в потрібному місці може автоматично зберегти та передати дані, коли користувач зайнятий виконанням інших задач [9].

Розпізнавання – потрібно опиратись на розпізнавання, а не на повторення. Підтримка інтерфейсом довгочасної пам’яті передбачає розпізнавання інформації користувачем, перш ніж він згадає її. Легше обрати якийсь об’єкт зі списку, ніж згадувати його правильну назву для введення в порожній рядок. Онлайнова допомога, повідомлення, поради

щодо користування інструментами, система контекстної допомоги тощо допомагають користувачу обирати інформацію, а не згадувати її.

Слід передбачити списки і меню, що містять об’єкти чи документи, які можна обирати, не змушуючи користувачів вводити інформацію в командному рядку без підтримки системи.

Інформування – наявність візуальних заставок. Необхідний аспект будь-якого графічного ІК та об'єктно-орієнтованого ІК полягає в тому, що користувачі повинні знати, в якому місці ПЗ вони знаходяться, які дії виконують наразі і які дії можуть виконати в подальшому.

Візуальна інформація служить нагадуванням для користувачів. Коли користувачі працюють в якомусь режимі або з мишею, це повинно відображатись на екрані за допомогою відповідної індикації. Форма курсора може змінюватись для вказання поточного режиму або дії, а індикатор – вмикатись і вимикатись. Візуальна інформативність продукту полягає в наявності візуальних підказок інтерфейсу, які інформують користувача про його поточні дії.

Терпимість – треба передбачити параметри за замовчуванням, команди відхилення (скасування) та повторення дії. Інтерфейс повинен надавати користувачу можливість зміни установок та налаштувань ПЗ, а також можливість повернення до установок за замовчуванням. Користувачі за своїми перевагами можуть змінити колір, шрифт, властивості прикладної програми або операційної системи, не задумуючись про їх початковий варіант. При розробленні інтерфейсу потрібно передбачити багаторівневі системи відхилення та повторення команд.

Швидкість – слід передбачити "швидкі" шляхи проходження інтерфейсу. Крім комбінованого використання миші та клавіатури існує ряд інших можливостей прискорити роботу з програмою. “Гарячі” клавіші та ярлики зменшують навантаження на пам’ять користувача та доводять виконання операцій до автоматизму.

Є 2 основних способи встановлення ярликів: прискорюючі та мнемонічні. Мнемонічні (або доступні) ярлики — це одиночні буквенно-цифрові символи, які встановлюють курсор на потрібний об’єкт та дозволяють зробити вибір. Вони використовують різні меню (панель, підменю, контекстні) та списки. Мнемонічні символи повинні бути унікальними для кожного роду дій. Наприклад, типове меню вікна використовує стандартні клавіші: F – для файлу, E – для редагування, V – для перегляду, H – для виклику довідкової системи. Наступний рівень меню (підменю) використовують свої налагодження мнемонічних клавіш: N – новий документ, O – відкрити, C – закрити, S – зберегти. Мнемонічні символи прискорюють рух і вибір потрібного меню або списку.

Інтуїтивність – потрібно активізувати синтаксис дій з об'єктами. Об'єктно-орієнтований інтерфейс надає ряд переваг від використання об'єктно-орієнтованого синтаксису взаємодії. Користувачам не потрібно запам’ятовувати, яку дію можна виконати в певний момент часу для об’єкта. При виборі об'єкта користувач бачить у меню дії, які можуть бути виконані над об’єктом. Недопустимі дії виділяються сірим кольором.

Об'єктно-орієнтований синтаксис дозволяє користувачу зрозуміти взаємозв’язок між об’єктами та діями в програмному продукті. В інтерфейсі слід використовувати метафори (переноси, які використовують назву об'єкту одного класу для опису об'єкту іншого класу) реального світу, які дозволяють користувачу переносити свої знання з реального світу у світ комп’ютерів. Наприклад, для програми Калькулятор не варто розробляти новий інтерфейс, він повинен співпадати з інтерфейсом звичайного калькулятора, який добре засвоїли користувачі. Однак при виборі та використанні метафор для інтерфейсу слід бути обережними. Обираючи метафору, потрібно зафіксувати її та слідувати їй весь час. Якщо виявиться, що метафора не відповідає своєму призначенню в усьому інтерфейсі, з'явиться необхідність вибору нової метафори.

Контекст слід застосовувати розкриття та пояснення понять і дій. Користувачі не повинні вважати можливості ПЗ більшими чи іншими, ніж вони є насправді. На кожному рівні інтерфейсу не варто показувати абсолютно всі функції ПЗ, а лише ті, в яких є потреба саме на даному рівні. Деякі програмні продукти надають різні меню для користувачів. Спочатку можна обрати просте меню для щоденних, звичайних потреб. Пізніше, по мірі засвоєння продукту або при виникненні потреби в більш складних властивостях програми, користувачі зможуть обрати більш складне меню. Це приклад того, як користувач одержує контроль над задачею та програмою.

2.2. Огляд існуючих методів аналізу користувацької статистики

Питання аналізу користувацьких дій на даний час є дуже актуальним. На сьогодні існує велика кількість методів аналізу користувацьких дій, їх кількість зростає з розвитком інформаційних технологій.

Одним з цих методів є діалог з користувачем у середовищі використання. Даний метод є застарілим оскільки його суть полягає в тому, що аналізується побажання кожного користувача індивідуально, що займає велику кількість часу для збору та організації отриманої інформації.

Ще одним застарілим методом, який протягом тривалого часу був популярний – це метод розробки дизайну додатку або середовища роботи безпосередньо опитуваних респондентів. Даний метод є теж доволі довгим для провадження але надає дуже велику кількість інформації для аналізу. Аналіз такої інформації займає дуже великий проміжок часу, проте дає дуже велику точність аналізованих даних. Ця точність досягається за рахунок співставлення побажання великої кількості людей, що зайняті в одній сфері. На сьогодні такий метод майже не користуються [10].

Метод аналізу фокус-групи є тим самим методом, від якого походить робота більшості працівників, які використовують додатки для аналізу

користувацької статистики сучасних комерційних в сервісах. Даний метод полягає в тому, що велика кількість людей, що зайняті в одній сфері. тестують додаток, який їм пропонують і на основі отриманих результатів власники продукту роблять висновки про його подальше вдосконалення. Недоліком такого додатку є те, що потрібно самостійно аналізувати та робити висновки з результатів, які були отримані під час тестування. По суті робота людей, які займаються аналізом інформації, є тією ж самою, що й робота працівників, які досліджують користувацьку статистику за допомогою різних комерційних додатків на встановлених веб-сайтах на сьогоднішній день. Їх робота є не систематизовано та неорганізованого, що виливається в те, що їх робота є не завжди ефективно.

Однак метод фокус-групи орієнтується на масовість і добре визначає поведінку більшості користувачів. В сучасному світі майже будь-який бізнес, що впливає на різні сфери діяльності, орієнтується на масовість. Це означає, що орієнтиром для вдалої реалізації продукту є більша кількість користувачів незалежно від того, в якій сфері діяльності або якого достатку є користувачі. Також цей метод має під собою і математично-статистичну базу, оскільки він є ілюстрацією нормального розподілу активності та побажань серед користувачів. Це означає, що за еталон поведінки користування додатком береться формат поведінки, який виконується більшістю користувачів. Згідно з нормальним розподілом, відхилення від формату такої поведінки буде існувати, але зустрічатися буде менше. Чим рідше воно буде зустрічатися, тим серйозніше буде відхилення.

Ще одним методом, який хотілося б зазначити, є немодероване віддалене тестування. Програмний продукт дають на тестування людям, які були обрані за різними соціальними та фінансовими показниками. Останні, в свою чергу, тестують додаток і під час тестування озвучують усі думки та дії, які вони виконують. Даний метод в основі своїй має психологічний аналіз, оскільки аналізуються соціальна поведінка

користувача а також його логічні дії. Недоліком такого методу як і в усіх попередніх методів є те, що аналізувати таку інформацію доволі важко. Такий аналіз є неорганізованим та неформалізованих та не завжди ефективним.

Контент-аналіз – якісно-кількісний метод вивчення документів, який характеризується об'єктивністю висновків і строгістю процедури та полягає у квантифікаційній обробці тексту з подальшою інтерпретацією результатів. Предметом контент-аналізу можуть бути як проблеми соціальної дійсності, котрі висловлюються чи, навпаки, приховуються у документах, так і внутрішні закономірності самого об'єкта дослідження. Популярність контент-аналізу ґрунтується на тому, що цей метод дозволяє виміряти людську поведінку (якщо вважати, що вербальна поведінка є її формою). На відміну від опитувань, контент-аналіз вимірює не те, що люди говорять, що зробили чи зроблять, а те, що вони справді зробили.

Може використовуватися як основний метод дослідження (наприклад, контент-аналіз тексту при дослідженні політичної спрямованості газети), в поєднанні з іншими методами (наприклад, в дослідженні ефективності функціонування засобів масової інформації), допоміжний або контрольний (наприклад, при класифікації відповідей на відкриті запитання анкет).

Виділяють два основних типи контент-аналізу: кількісний і якісний. Якщо кількісний аналіз націлений на виявлення частоти окремих тем, слів або символів, що містяться у тексті, то якісний аналіз пов'язаний з фіксуванням нетривіальних висловлювань, мовних інтонацій з розумінням цінності змісту повідомлення [11].

Одним з найважливіших методів аналізу тестування користувацької статистики є спліт-аналіз. Це означає, що береться вибірка користувачів, яким надається один і той же продукт декілька разів проте з кожним разом в продукті змінюється якась одна особливість. Наприклад, якщо користувач тестує якийсь окремий веб-сайт або веб-сторінку, то можуть

бути видозмінені або переставлені місцями деякі елементи інтерфейсу. Такий метод є інтерактивним і займає велику кількість часу, проте надає серед усіх вищезазначених методів найбільшу точність аналізованої інформації. Однак його недоліком є те, що, як і у всіх попередніх методах, даний метод не пропонує жодних висновків з отриманої інформації.

Як бачимо, що в усіх існуючих методах аналізу користувацької активності існує проблема аналізу висновків. Жоден з цих методів не пропонує жодних висновків для того, щоб визначити, які дії потрібно здійснити, щоб досягнути бажаної цілі.

2.3. Формулювання запропонованого програмного методу

Основною відмінністю запропонованого методу є те, ще він надає висновки по зібраній інформації. За основу взято метод фокус-груп. Це означає, що веб-сайт, на який встановлюється розроблюваний додаток, буде збирати статистику користувацьких дій, як і в методі фокус груп. Але після збору статистики буде розпочинатися робота аналізу зібраних даних.

Аналізуватися буде кожен елемент інтерфейсу. Для опису методу розглянемо приклад: інтернет-магазин. Всі інтернет-магазини мають в цілому велику кількість спільних особливостей. Вони всі мають каталог товарів, сторінку для кожного товару, і на цих сторінках існують різні фільтри. Для того, щоб оцінити вдалість розташування елементів інтерфейсу, ми будемо відштовхуватися від загального алгоритму дій, який виглядає наступним чином:

- збір кількості кліків по кожному елементу інтерфейсу;
- розбиття елементів на 2 головні групи;
- оцінка важливості елементу в кожні группі: (у відсотках);
- створення таблиці, що сортується за зростанням важливості, де показано середній час пошуку елементу;
- виділення елементів в таблиці, що погано розташовані.

Інформація по клікам користувачів містить в собі наступні дані: кількість секунд від попередньої дії користувача, яка знадобилася йому для взаємодії з даним елементом, та власне елемент, з яким була здійснена взаємодія. Під елементом мається на увазі будь-яка частина веб-інтерфейсу: кнопка, фільтр, радіо-батон, зображення тощо. Узагальнюючи вище написане, можемо сказати, що це будь-що, що при кліку по собі надсилає пакети на сервер.

Згідно правилу нормального розподілу, серед усіх елементів інтерфейсу веб-сайту будуть ті, що клікаються найчайстіше, та ті, що майже ніколи не натискаються. З часом, при поповненні банку даних користувацьких дій, можна зробити висновки.

Перший: найбільш часто шуканий елемент повинен мати найменший час пошуку. Тому якщо елемент натискається найчастіше, проте має не найменший час пошуку, це означає що його місце в інтерфейсі потрібно змінити.

Другий: всі елементи можна розділити на 2 категорії, “більш важливі” та “менш важливі”. Перші при натиснені збільшують довжину URL, по якому переходить користувач. Другі, навпаки зменшують. Очевидно що при збільшенні довжини адреси сторінки, на яку переходить користувач, спостерігається його зацікавленість у веб-сайті. Як приклад можна навести те, що при виборі категорії товарів в інтернет-магазині довжина URL збільшується, і при виборі фільтрів в даній категорії теж. Елементи, які переводять користувача по таким посиланням і є більш важливими. До менш можливих можна віднести типові вкладки будь-якого веб-сайту, такі як “Реєстрація”, “Вхід до особистого кабінету”, “Контакти” тощо.

В минулому веб-сайти створювали навмисне складними, для того, щоб користувач на довше на них залишався. Проте з часом веб-сайтів стало неймовірно багато, і користувач, на сьогодні, в разі невподобання веб-сайту, покидає його. Проте питання подовження середнього часу сесії

користувача є найбільш важливим і на сьогодні. Згідно з дослідженням Google, було виявлено, що більшість успішних сайтів подовжують середню тривалість сесії користувача за рахунок того, що користувач швидше знаходить необхідні йому предмети на веб-сайті, проте під-час їх пошуку проходить більшу кількість веб-сторінок. Отже, в разі правильного розташування елементів інтерфейсу веб-сайту, користувач буде постійно переходити по посиланням, до поки не досягне своєї цілі, що як наслідок, збільшить середній час користувацької сесії [12].

2.4. Висновки

В даному розділі було розглянуто основні аналоги веб-сервісів, що надають статистику по користувацьким діям на веб-сайті на якому були встановлені. Основним недоліком більшості з них є те, що такі додатки потрібно створювати окремо під кожну платформу. Окрему увагу слід звернути на додаток Google Analytics, який в своїх функціональних можливостях використовує створенні компанією, що його розробляла продукти. Перевагою переглянутих додатків є великий обсяг статистичної інформації яку можна відстежувати, а також зручний графічний вигляд в якому вона подається.

Основним недоліком у всіх вище переглянутих додатків є те, що жоден з них не пропонує аналізу та висновків зібраної статистики. В ході ведення бізнесу це спричиняє до того що власник веб-сайту повинен наймати окремих працівників, що будуть аналізувати статистикою надають дані додатки. Робота цих працівників й несистематизованою та неорганізованою, що означає, що кошти, що витрачаються на таких працівників є малоефективними.

3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МЕТОДУ

3.1. Опис засобів розробки програмного забезпечення

В ході виконання роботи були використані різні технології та мови програмування. Однією з них є мова javascript та фреймворк React.js.

JavaScript має С-подібний синтаксис, але в порівнянні з мовою С має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;
- функції як об'єкти першого класу;
- обробка винятків;
- автоматичне приведення типів;
- автоматичне збирання сміття;
- анонімні функції.

React.js було обрано через його простоту та швидкість написання застосунків за допомогою нього. Розроблений додаток встановлюється на веб-сайті і взаємодії з інтерфейсом обраного веб-сайту. Javascript є високорівневою мовою що була створена на початку XXI століття. Дана мова була створена для того щоб оживити сторінку веб-сервісів. На сьогодні за допомогою javascript можна створювати повноцінні веб-додатки.

З кожним роком зростає кількість бібліотек напрямків для нього що збільшують напрями використання даної мови програмування. Javascript підтримує принципи об'єктно-орієнтованого програмування. Також для даної мови програмування існують механізми для взаємодії з різними базами даних, як реляційними (наприклад mysql, postgresql) так і не реляційними (наприклад mongodb), а також базами даними кешування наприклад Redis. В даній роботі була використана бібліотека AJAX оскільки більше частина коду виконується асинхронно від дій

користувачів AJAX надає можливість посилати на сервер пакети не перезавантажуючи сторінку користувача. Оскільки розроблюваний додаток взаємодіє з сайтами використання даної мови та бібліотек були необхідними для його створення. Для швидкої обробки даних на сервері а саме даних які були зібрані з користувацьких дії було вирішено створити модуль на морі C++.

C++ є наступним етапом розвитку класичної мови Сі і по суті являє собою мову Сі в якій вбудовано механізми для програмування згідно з парадигм. Нововведеннями C++ порівняно з C є:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- простори імен;
- вбудовані функції;
- перевантаження операторів;
- перевантаження імен функцій;
- посилання і оператори управління вільно розподіленою пам'яттю.

Дана мова підтримує інкапсуляцію, поліморфізм та наслідування Великим плюсом даної мови є те що вона має ряд технологій які підтримують взаємодію даної мови з іншими технологіями, наприклад javascript, C# та інші. В даній роботі мова C++ була використана лише для обробки статистичної інформації і даний модуль виконує обробку та аналіз інформації. Оскільки окрім веб додатку було створено настільний застосунок під платформу Windows, для взаємодії з ним було використано технологію C++/CLI. Дана технологія дозволяє створити взаємодію компонентів що написані на мові Сі-Шарп з модулями C++. Основи механізму взаємодії даних двох модулів є принцип побудови делегатів в

коді. Це означає що в обох моделях було створено класи з великою кількістю методів які присутні як в модулі написаному на C# так і в модулі написаному на мові C++ [13].

Для збереження даних було використано нереляційну базу даних MongoDB. Вибір нереляційної бази даних переваги через те, що оскільки кількість користувацьких метрик та їх характер можуть змінюватися під-час розробки, то потрібно уникати прив'язаності до строго визначеного порядку та формату даних, які будуть циркулювати між модулями застосунку.

Оскільки для демонстрації роботи додатку потрібно створити веб-сайт, потрібно обрати мову програмування, що має можливість роботи з фреймворком, який зможе надати середовище та можливості для зручної роботи з реляційною базою даних, а також для створення шаблону проетування MVC [14].

Python – високорівнева мова програмування, що має широкий вибір призначень, в яких її можна використовувати. Одна з головних особливостей даної мови – можливість швидкого написання коду, що підвищує продуктивність розробника. Стандратна бібліотека містить в собі великий обсяг корисних функцій для роботи. Варто зазначити, що Python підтримує різні парадигми програмування, зокрема структурне, об'єктно-орієнтоване, процедурне, імперативне та аспектно-орієнтоване. На даний момент широко використовують дві версії мови прорамування, а саме Python 2.7 та Python 3.6. Кожна з них має свої певні особливості.

Дана мова працює на майже всіх існуючих платформах. Щодо типів та структур даних слід зазначити, що Python підтримує динамічну типізацію, отже тип даних змінної в коді взначається лише під час виконання коду. Дана мова має наступні вбудовані типи даних: логічний булвевий, строковий, Unicode-строковий, чисельні типи та інші. Створити новий тип даних можна реалізувавши його за допомогою нового class

об’єкту. Класи в Python підтримують наслідування, як одиночне так і множинне [15].

Як і будь-яка мова програмування, Python має свої переваги та недоліки, які можуть критично вплинути на вибір технологій перед початком розробки. Python має низьку швидкість виконання в порівнянні з C/C++, проте якщо порівнювати дану мову з її альтернативами, а саме PHP, JavaScript, Ruby, то можна впевнено сказати, що серед даних чотирьох технологій [16].

Python має найкращий показник виконання коду. Основні переваги Python:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

Javascript – мова програмування, що може використовуватись в таких парадигмах програмування, як об’єктно-орієнтоване, процедурне,

імперативне, і також є високорівневою мовою програмування з великою кількістю додаткових бібліотек.

JavaScript має С-подібний синтаксис, але в порівнянні з мовою С має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через
- механізм прототипів;
- функції як об'єкти першого класу;
- обробка винятків;
- автоматичне приведення типів;
- автоматичне прибирання сміття;
- анонімні функції.

Наразі дана мова використовується для програмного доступу до об’єктів додатку, з яким працює. Історично JavaScript був створений для покращення вигляду веб-сторінок, і використовувався як мова сценріїв, для створення інтерактивного вигляду веб-сторінок. На сьогодні існує багато фреймворків, що використовують дану мову програмування, а саме AngularJS, ReactJS, Node.js та інші. JavaScript має засоби для створення як клієнтської, так і серверної частини веб-сайтів. Проте використання пов’язаних з цим технологій займає велику кількість часу [17].

Мова JavaScript використовується для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових веб-додатків (ReactJS, AngularJS, Vue.js);
- програмування на стороні сервера (Node.js);
- стаціонарних додатків (Electron, NW.js);
- мобільних додатків (React Native, Cordova);
- сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite).

В даному проекті JavaScript буде використовуватись для створення інтерактивності сторінок веб-сайту, а також для підтримки технології AJAX, що виконана за допомогою бібліотеки jQuery. Основним використанням данної мови буде створення настільного додатку з адміністраторським інтерфейсом, використовуючи фреймворк Electron.

Так як для демонстрації розроблюваного додатку необхідно створити веб-сайт, на якому буде демонструватися його робота, при розробці буде використовуватись мова розмітки веб-сторінок HTML. HTML не є мовою програмування, а лише є мовою, завдяки якій створюється розмітка та вигляд веб-сторінок. Дана мова інтерпретується браузерами, і саме в результаті інтерпретації веб-сторінка претворюється з тексту, на ті елементи, які були створені. В цілому HTML – тегова мова розмітки документів. Будь-який документ, написаний даною мовою, являє собой набір певних елементів, при чому початок та кінець цих елементів описуються спеціальними структурами – тегами [18].

HTML впроваджує засоби для:

- створення структурованого документа шляхом позначення
- структурного складу тексту: заголовки, абзаци, списки, таблиці,
- цитати та інше;
- отримання інформації із Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до тексту.

SQLite – це реляційна СКБД. Навідміну від більшості СКБД SQLite не може звертатися до веб-джерел, тому всі запити є запитами до пам’яті на жорсткому диску. Простота реалізації даної технології полягає в тому, що перед початком виконання будь-якої операції, весь файл з базою даних блокується, по черзі оброблюючи всі вхідні запити.

Оскільки метою дипломного проекту є не створення веб-сервісу, а створення додатку для моніторингу та онлайн-консультування користувачів, то створюваний тестовий веб-сайт не потребує СКБД, що може витримувати великі навантаження.

MySQL – вільна реляційна СКБД, що розроблюється та підтримується корпорацією «Oracle». MySQL найчастіше використовується в малих та середніх додатках в якості сервера, до якого звертаються локальні або віддалені клієнти. На даний момент дана СКБД портована на велику кількість платформ, та має API для роботи на великій кількості мов програмування, зокрема на Python, PHP.

MySQL легка СКБД, проте розробка її ядра займає доволі довгий час, і хоч для неї існує велика кількість графічних редакторів, що дозволяють швидко та зручно із нею працювати, розробка веб-сайт з її використанням буде відбуватися повільніше [19].

Redis – мережеве сховище даних, що автоматично журналюється, типу «ключ-значення», з відкритим сирцевим кодом, що являє собою нереляційну високошвидкіну СКБД. Redis зберігає базу даних в оперативній пам’яті, але також має механізми для постійного збереженення даних на диску. Дана СКБД працює на більшості POSIX систем, таких як Linux, BSD, Mac OS X. Офіційної підтримки платформи Windows наразі не існує. Для управління даними підтримуються такі команди, як інкремент/декремент, стандартні операції над списками і множинами (об'єднання, перетин), перейменування ключів, множинні вибірки та функції сортування. Підтримується два режими зберігання: періодична синхронізація даних на диск і ведення на диску логу змін. У другому випадку гарантується повне збереження всіх змін. Можлива організація master-slave реплікації даних на кілька серверів, здійснювана в неблокуючому режимі. Доступний також режим обміну повідомленнями "публікація/підписка", при якому створюється канал, повідомлення з якого поширюються клієнтам за передплатою [20].

В даному веб-сайті, окрім SQLite, було використано Redis для здійснення кешування повторюваних запитів. Оскільки для веб-сайтів, що пов’язані з продажем будь-чого, повторювані запити – явище постійне, було доцільно здійснити кешування даних. Таким чином, при зростанні кількості запитів до СКБД, і при збільшенні повторюваних запитів, час обробки запиту не буде рости лінійно [21].

3.2. Опис модулів розробленого застосунку

Загальна архітектура додатку представлена на рис. 8. В ній можна виділити 3 головні модулі:

- модуль збирання статистичних даних;
- модуль обробки статистичних даних;
- адміністраторський модуль.

Першим було створено модуль, що збирає інформацію про користувацьку активність.

Оскільки в задачі на проект було визначено, що додаток буде встановлюватись на веб-сервісах, що написані за допомогою React.JS, даний модуль було оформлено у вигляді бібліотеки JavaScript.

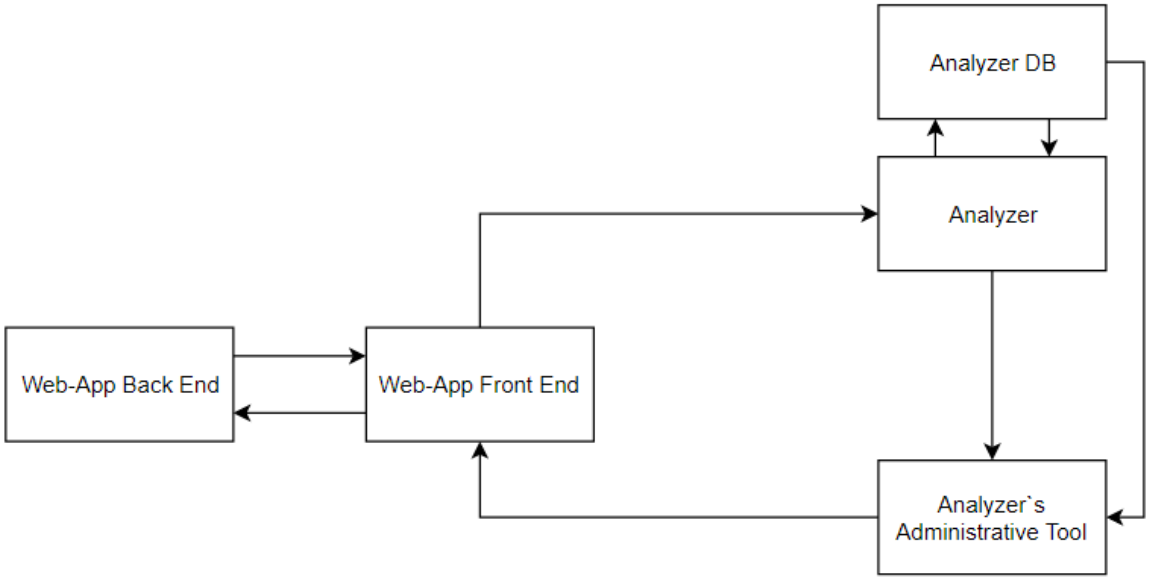


Рис. 8. Блок-схема взаємодії компонентів додатку

Основні функції модуля збирання статистики:

- збирання інформації по клікам кожного користувача по кожній веб-сторінці;
- збирання інформації по клікам користувача по кожному елементу інтерфейсу кожної веб-сторінки;
- замір часу сесії користувача;
- замір часу перебування користувача на кожній веб-сторінці;
- замір часу взаємодії з елементами інтерфейсу;
- передача всієї зібраної інформації до модуля обробки користувацької статистики.

Даний модуль використовує бібліотеку AJAX. AJAX є одним з підходів до розробки користувацьких веб-інтерфейсів, який передбачає «фоновий» обмін даними між браузером і серверними ресурсами. Асинхронний спосіб оновлення даних веб-сторінок дозволяє не перезавантажувати їх вміст повністю. Завдяки цьому веб-ресурс працює більш плавно і швидко. Також це дозволяє економити трафік, що особливо актуально в умовах, якщо немає безлімітного інтернету [22].

Але коли на сайті застосовується технологія AJAX, користувачі помічають часткову зміну сторінок. Тому розробникам слід подумати над впровадженням індикації цих змін. Користувач повинен розуміти, що зараз відбувається фоновий обмін даних з сервером.

Ще одна важлива особливість полягає в тому, що не всі браузери працюють з AJAX. Його не підтримують, наприклад, старі і текстові версії браузерів. А ще іноді Javascript блокується користувацькими налаштуваннями. Тому розробники часто рекомендують шукати для технології більш гнучкі альтернативи та інші методи відображення даних на веб-ресурсі [24].

AJAX не завжди рекомендується застосовувати розробникам. В окремих випадках технологія принесе свої плюси, але іноді може погано позначитися на роботі веб-ресурсу.

До основних переваг підходу належать:

- зменшення трафіку – завдяки частковому оновленню ресурсів не потрібно перезавантажувати сторінку;
- зниження навантаження на сервер – технологія дозволяє робити менше запитів до бази даних;
- швидка робота сторінок сайту – відгук на дії користувача відбувається більш динамічно;
- поліпшення функціональності веб-ресурсу.

Оскільки описаний модуль перевантажує обробку всіх GET/POST запитів, дана бібліотека знайшла продуктивне використання в розробленому додатку.

Зібрана інформація користувацьких дій відправляється в модуль обробки статистичної інформації. Основні функції даного модуля:

- збереження всієї зібраної інформації, що попередньо групується;
- оцінка загальних метрик по кожній веб-сторінці;
- оцінка якості розташування елементу в інтерфейсі;
- передача висновків до адміністраторського модуля веб-додатку та настільного додатку.

Адміністраторські модулі створені для відображення отриманої інформації в читабельному вигляді.

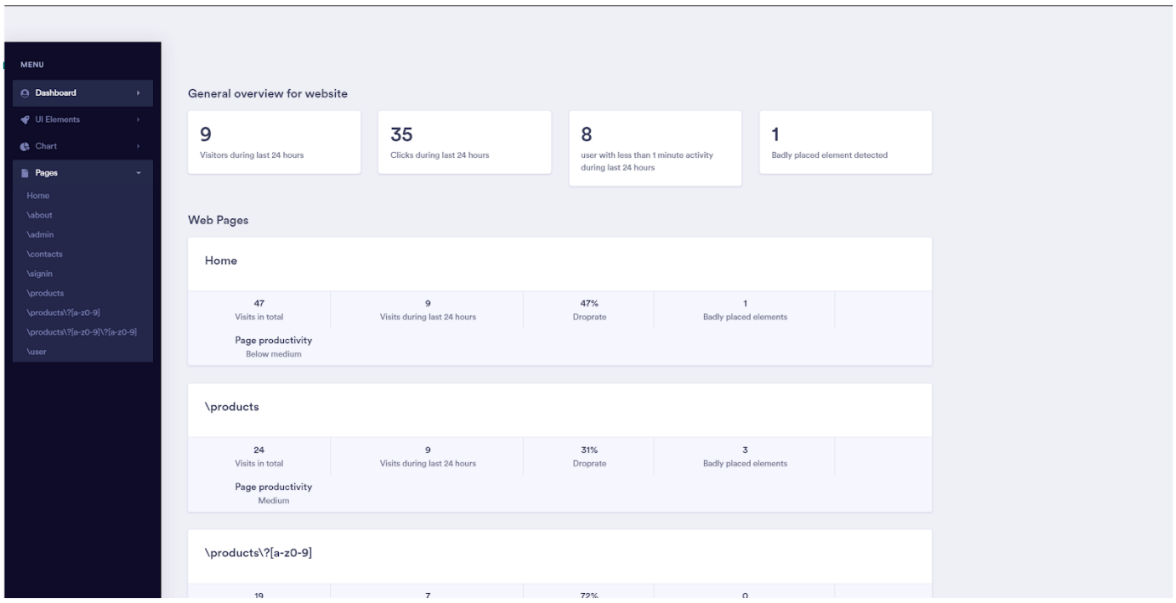


Рис. 9. Приклад роботи розробленого додатку (головне меню)

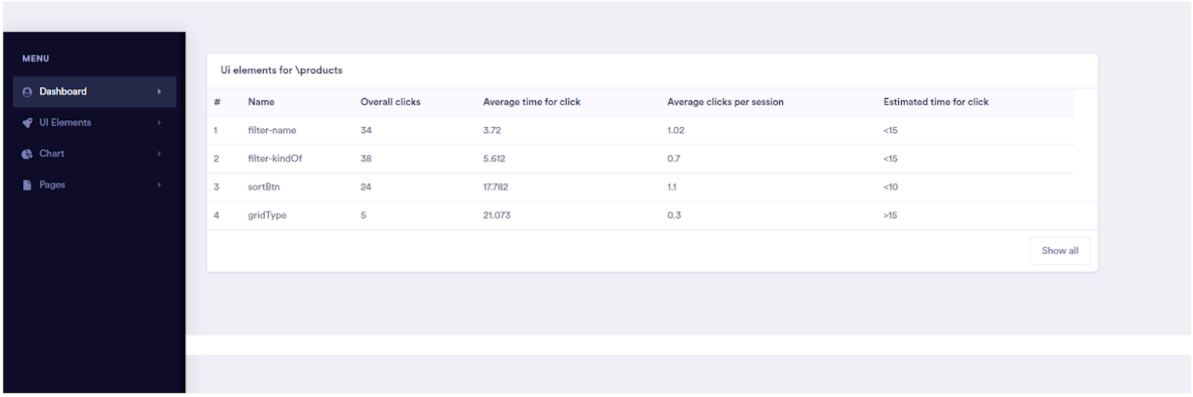


Рис. 10. Приклад роботи розробленого додатку (меню інформації по сторінці)

3.3. Висновки

В даному розділі було розглянуто основні аналоги веб-сервісів, що надають статистику по користувацьким діям на веб-сайті на якому були встановлені. Основним недоліком більшості з них є те, що такі додатки потрібно створювати окремо під кожну платформу. Окрему увагу слід звернути на додаток Google Analytics, який в своїх функціональних можливостях використовує створенні компанією, що його розробляла продукти. Перевагою переглянутих додатків є великий обсяг статистичної

інформації яку можна відстежувати, а також зручний графічний вигляд в якому вона подається.

Основним недоліком у всіх вище переглянутих додатків є те, що жоден з них не пропонує аналізу та висновків зібраної статистики. В ході ведення бізнесу це спричиняє до того що власник веб-сайту повинен наймати окремих працівників, що будуть аналізувати статистикою надають дані додатки. Робота цих працівників й несистематизованою та неорганізованою, що означає, що кошти, що витрачаються на таких працівників є малоефективними.

4. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1. Методика оцінювання отриманих результатів

В ході тестування та аналізу результату ефективності програмного забезпечення було здійснено наступні дії. Вибірці з 70 користувачів було запропоновано для користування 5 комерційних веб-додатків, front end частина яких була написана за допомогою React.JS. Користування проходило в 2 етапи. На першому етапі веб-сайт ніяк не змінювався, але на ньому вже було встановлено розроблений додаток.

Перший етап збирав банк даних для аналізу. Отримавши результат, було здійснено зміни в інтерфейсі веб-сайтів, що були запропоновані користувачам, після чого користувачам знову було запропоновано до користування ті самі веб-сайти, але в яких було внесено зміни.

4.2. Оцінка отриманих результатів

Отримані результати зображено в табл. 1.

Таблиця 1

Оцінка отриманих результатів

№ сайту	Показник ASD (average session duration) до/після (в секундах)	Показник AVP (average visited pages) до/після (в секундах)	Показник ADP (average drop rate) (%)
1	134.67/143.84	4.67/7.31	46.22/42.73
2	111.2/138.56	3.54/4.78	41.25/39.64
3	67.34/71.61	3.78/5.05	78.51/65.86

Продовження табл. 1

4	145.88/211.34	6.98/11.23	34.22/11.57
5	78.14/88.97	3.83/4.26	56.22/41.27

4.3. Висновки

В ході аналізу результатів було отримано задовільні показники. Розроблений додаток було протестовано на п’яти різних веб-сайтах. Основними критеріями аналізу успішності розробленого додатку було обрано три метрики:

- середній час користувацької сесії;
- середня кількість відвіданих сторінок;
- середній відсоток користувачів, що покинули веб-сайт не проглянувши більше 3 сторінок за час більший 30 секунд.

Отримані результати показали, що при внесенні змін, згідно з отриманою інформацією, яку надає розроблений додаток, показують, що розроблений програмний метод добре себе продемонстрував на всіх п’яти веб-сайтах, поліпшивши усі метрики.

В середньому були отримані такі результати:

- Average droprate знизився на 15%.
- Average session duration зросла на 23%.
- Average amount of visited pages зросла на 14%.

5. ПОБУДОВА БІЗНЕС-МОДЕЛІ

5.1. Опис проблеми

На сьогодні, велика кількість комерційних операцій як в світі, так і в Україні, здійснюються за допомогою електронних джерел. Електронні джерела використовуються в державних органах врядування, податкових інспекціях тощо.

Підприємці, створюючи нові комерційні веб-сервіси, зіштовхуються з багатьма проблемами, такими як створення інтуїтивно зрозумілих інтерфейсів, повнота функціональних можливостей, загальне підвищення якості свого застосунку, підвищення якості праці власного персоналу. Здійснення даних проблем реалізовується на основі аналізу статистичних даних. Саме тому в останній в світі почали з’являтися різні мікросервіси та надбудови над веб-застосунками, що збирають різну статистичну інформацію та допомагають покращити взаємодію з клієнтами. Такі сервіси надають такі функції, як: вбудова на сайт чату, засоби спостереження за діями відвідувача, створення мапи кліків відвідувачів, відстеження звідки користувач дізнався про Ваш сайт тощо.

Таким чином маємо ряд застосунків, які створені з єдиною метою – покращити роботу комерційного веб-застосунку. Однак проблема всіх цих сервісів полягає в їх роздрібленості: не має жодного рішення, яке б уособлювало в собі весь функціонал, що відповідав би вирішенню вище заданих проблем. Саме тому варто звернути увагу на створення такого застосунку.

Також великий вплив на комерційну успішність будь-якого веб-сервісу здійснює його зовнішнє оформлення – інтерфейс та дизайн. Найпоширеніші шаблони інтерфейсів створюються внаслідок наслідування зовнішнього вигляду високобюджетних успішних застосунків. Як приклад, одним з найбільших інтернет-магазинів України є інтернет-магазин “Розетка”. І в Україні більшість інтернет магазинів

намагаються скопіювати його зовнішній вигляд, змінюючи лише дизайнерські деталі, такі як шрифти, кольорова гама тощо. Це означає, що інтерфейс веб-застосунку можна формалізувати, а формалізовану модель можна аналізувати на предмет різних характеристик та метрик, які зможуть за рахунок відслідковування користувацької статистики покращити зовнішній вигляд веб-застосунку [23].

Мапа кліків користувачів, а також відслідковування часу між кліками користувача на веб-сайті може надати інформацію про те, які елементи інтерфейсу розташовані невдало. Наприклад якщо найчастіше шуканий користувачами елемент інтерфейсу має найдовший час пошуку в порівнянні з іншими елементами, то це означає, що даний елемент має бути відображений на найбільш явному місці веб-сторінки.

Все вище описане можна відобразити у схемі “Дерева проблем”, що зображено на рис. 11.

5.2. Зацікавлені сторони

У вирішенні вище розглянутої проблеми існує декілька зацікавлених сторін. Найбільшою зацікавленою стороною є малий та середній бізнес, що для своєї реалізації використовує веб-сервіси, тобто будь-що, що може мати веб-сайт для зв’язку з клієнтами. Іншою зацікавленою стороною є користувач, оскільки глобальне покращення якості інтерфейсу веб-застосунку скорочує час користувача, що витрачається на пошук бажаної послуги, товару тощо.

Оскільки більшість веб-сервісів для малого та середнього бізнесу створюється IT компаніями, то вони також являють собою зацікавлену сторону, оскільки даний застосунок збільшить швидкість формулювання та зрозумілість поставлених перед ними задач, що в свою чергу збільшує продуктивність IT компаній.

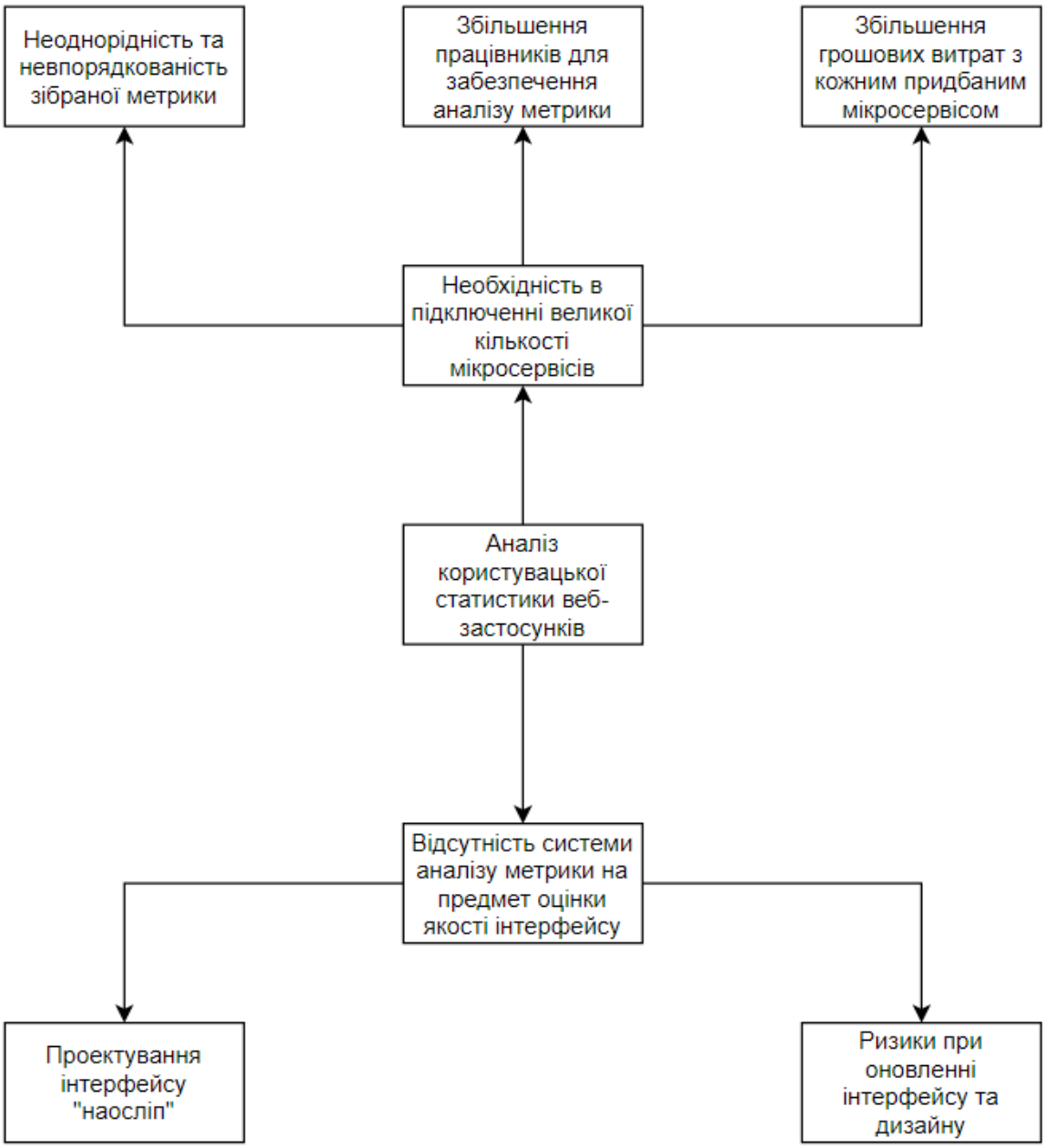


Рис. 11. Дерево проблем

В табл. 2 зведено всі групи зацікавлених сторін, їх інтереси та впливи (міра зацікавленості у вирішенні наявних проблем).

Таблиця 2

Зацікавлені сторони

Зацікавлена сторона	**Інтерес зацікавленої сторони**	**Вплив зацікавленої сторони**	**Стратегії приваблення зацікавлених сторін**
Бізнес	Збільшення прибутку за допомогою метрики власного веб-сервісу. Підвищення автоматизації роботи відділів, що взаємодіють з клієнтами	Високий	Участь у спеціалізованих виставках, форумах, конференціях, проведення презентацій для зацікавлених компаній
Клієнти	Скорочення часу пошуку бажаної послуги, товару тощо	Низький	
ІТ компанії	Збільшення час на виконання проектів	Низький	

5.3. Комерційне рішення. Основні характеристики

Відповідно до вищезазначених проблем можна описати кінцевий продукт, що буде їх вирішувати. Даний продукт буде реалізовувати інтерфейс для підтримки контакту між клієнтами та адміністраторами, а також буде збирати та аналізувати метрику користувацької статистики на кожній сторінці веб-сервісу по кожному елементу інтерфейсу, відповідно до вимог і за допомогою алгоритмів, описаних у попередніх розділах. Результат даного застосунку буде видно через деякий час, після його встановлення оскільки потрібно спочатку зібрати банк користувацької активності з моменту його встановлення.

По-перше, зібрана статистика метрики користувацької активності буде представлена і зручно читабельному форматі, що виключає час відповідного співробітника на аналіз метрики. Це означає, що процес аналітики користувацької активності буде більш автоматизованим, що в

свою чергу зменшує кількість витрат на співробітників. По друге, даний сервіс міститиме в собі інші додаткові можливості для підтримки контакту з клієнтами, такі як автоматизовані чат-боти, що в свою чергу збільшить час реагування на клієнтські запити, та зменшить витрати на співробітників, що спеціалізуються на контакті з клієнтами.

З цього випливає, що власники комерційних веб-сервісів є клієнтами даного сервісу, а співпраця буде побудована на моделі B2B. Даний програмний продукт повинен бути зрозумілим у використанні та встановленні, надавати всю необхідну інформація для співробітників веб-сервісу, на якому будуть встановлюватися.

5.4. Конкурентні переваги рішення

У попередніх розділах було проаналізовано ключові недоліки конкурентних застосунків. Найбільшим з них є роздрібленість функціональних можливостей застосунку: не існує на даний момент застосунку, що містив би в собі можливості збору різносторонньої користувацької активності, а ті, що виконують збір певної користувацької активності, не завжди подають її результат в зручному вигляді.

Головною різницею розроблюваного застосунку є збір користувацької статистики не лише по кожній сторінці веб-сервісу, але й по кожному елементу інтерфейсу веб-сторінки. Така особливість відкриває великі можливості аналізу роботи і проектування інтерфейсу веб-застосунків, що збільшить точність зібраної інформації, і, як наслідок, збільшить цінність зібраної метрики для веб-додатку, на якому наш мікросервіс встановлено.

Отже, конкурентними перевагами є:

- зменшення кількості часу на аналіз зібраної метрики;
- збільшення точності отримуваної метрики;
- унікальний метод аналізу ефективності інтерфейсу.

5.5. Клієнти, сегменти ринку споживання

Для Досягнення максимальної ефективності діяльності команди проекту рекомендується провести сегментацію ринку клієнті, тобто виділити більш-менш однорідні групи користувачів. зацікавлених у пропонованому продукті.

Таким чином сегментацію ринку клієнтів для застосунку, що аналізує показники інтерфейсу веб-сервісів на основі користувацької статистики за наступними параметрами:

- обсяг очікуваних від’єднань певної веб-сторінки веб-застосунку;
- обсяг щоденних відвідувань веб-сайту.

Обидва критерії підходять для групи клієнтів, що представляють собою малий та середній бізнеси, що зацікавленні у відстежуванні і покращенні вищезазначених показників.

5.6. Унікальна ціннісна пропозиція

Ціннісна пропозиція – це пояснення того, як продукт вирішує проблему. Його можна скласти за формулою:

Ціннісна пропозиція = Проблема + Рішення/Продукт.

У дереві проблем було виділено низку проблем, а у зацікавлених сторонах - очікування сторін від продукту.

Власники веб-сервісів планують отримати:

- засіб комунікації з клієнтами: чат-бот, що буде відповідати на найбільш поширені питання користувачів, що в подальшому зможе під’єднати оператора у разі, якщо питання користувача не

було вирішено. Модуль з чат ботом повинен бути легко модифіковним, щоб із збільшенням повторюваних питань їх можна було додати до тих, на які буде відповідати чат бот.

- загальна метрика веб-сайту: зручно читабельна статистика користувацької активності по всьому веб-сайту. Вивід зібраної метрики повинен підкріплюватися діаграмами та часовими графіками, мапами кліків.
- метрика кожної веб-сторінки: по кожній веб-сторінці веб-сайту повинен бути проведений аналіз користувацької активності по кожному елементу інтерфейсу, для виявлення погано розташованих елементів на веб-сторінці. Отже, останній пункт наведеного вище списку є унікальною пропозицією, яку на даний момент не було реалізовано в жодному аналогічному застосунку.

5.7. Доходи та витрати

Сумарний дохід являє суму продажів ліцензій на запропоноване ПЗ. Хоч даний застосунок працює з користувацькою активністю веб-сервісів, що означає, що для роботи даного застосунку потрібне підключення до мережі Інтернет, через складність алгоритму аналізу метрики по кожному елементу кожної веб-сторінки, він повинен мати доступ до сирцевого коду фронт-енд частини застосунку, на якому встановлюється. Монетизація буде відбуватися за рахунок продажу ліцензій на запропоноване програмне забезпечення. Пропонується продавати комерційну ліцензію. Таке вирішення було зроблено з міркувань безпеки збереження сирцевого код розроблюваного застосунку, а також з метою ризиків розповсюдження неліцензіованого варіанту розроблюваного програмного забезпечення. Оскільки комерційна ліцензія зберігає майнові права за власником, це захищає код пропонованого застосунку. У вартість такої ліцензії входить

технічна підтрика клієнтів на той період часу, на який була куплена ліцензія.

До витрат можна віднести наступні пункти:

- утримання персоналу для надання технічної підтримки (виплати заробітних, соціальних плат);
- утримання робочих місць для персоналу.

Детальніше про витрати в табл. 3 та табл. 4.

Таблиця 3

Структура доходів за перше півріччя

Найменування витрат	**1-й місяць т. $**	**2-ий місяць т. $**	**3-ій місяць т. $**	**4-ий місяць т. $**	**5-ий місяць т. $**	**6-ий місяць т. $**
Загальні витрати	4	8	8	2	2	2
ЗП		20	20	20	20	20
Витрати	4	28	28	22	22	22
Заплановані прибутки						
Результат (без оподаткування)	–4	–28	–28	–22	–22	–22

Таблиця 4

Структура доходів за друге півріччя

Найменування витрат	**7-ий місяць т. $**	**8-ий місяць т. $**	**9-ий місяць т. $**	**10-ий місяць т. $**	**11-ий місяць т. $**	**12-ий місяць т. $**	∑
Загальні витрати	2	2	2	2	2	2	24

Продовження табл. 4

ЗП	20	20	20	20	20	20	220
Витрати	22	22	22	22	22	22	258
Прибутки	40	45	55	70	95	120	425
Результат	18	23	33	58	73	98	167

5.8. Бізнес-модель

Споживачі: компанії, що реалізують свої продажі/послуги за допомогою веб-сайтів.

Проблема: відсутність автоматизованих рішень; висока вартість людських послуг; низька якість послуг модераторів; недостатня точність метрики існуючих аналогів.

Рішення: програмне забезпечення, надає модуль для напівавтоматизованого зв’язку з клієнта, а також що вказує на недоліки спроектованого інтерфейсу на основі аналізу користувацької активності.

Унікальна ціннісна пропозиція: програмне забезпечення, що аналізує продуктивність створеного інтерфейсу веб-сайту на основі аналізу розташування кожного елементу інтерфейсу.

Потоки доходів: доходи від продажу ліцензій;

Структура витрат:

- утримання персоналу для надання технічної
- підтримки (виплати заробітних плат, соціальних виплат);
- утримання робочих місць для персоналу (оплата за оренду офісу та комунальні послуги);
- податкові витрати;
- оплата послуг юриста, бухгалтера, прибиральниці.

Також в канву бізнес-моделі включаються структурні блоки:

- прихована перевага (перевага, яку не можливо скопіювати або купити),
- ключові метрики (основні показники, що вимірюються) та канали (шляхи до користувачів).
- Канали: через відділи співпраці/інтеграції відповідних мікросервісів.
- Ключові метрики: кількість проданих ліцензій.

Прихована перевага: врахування значення важливості веб-сторінки для даного веб-сайту. Бізнес-модель наведена у зведеному вигляді в табл. 5. Отже, зважаючи на табл. 5, можна зробити висновок, що запропонований проект, який реалізує описаний у дисертацій метод аналізу продуктивності інтерфейсу на основі відстеження користувацьких дій, має перспективи у своїй подальшій реалізації. Звичайно, проведений аналіз не враховує всіх ризиків та факторів, таких як специфіка оподаткування у країні ведення бізнесу, проте навіть наявних досліджень достатньо, щоб прогнозувати комерційний успіх продукту та його окупаємість.

5.9. Висновки

У даному розділі було проведено аналіз поточної ситуації у сфері збору метрики користувацької метрики на веб-сайтах, виявлено наявні проблеми та підсумовано їх у відповідному дереві проблем. Наряду з проблемами було виділено основні зацікавлені сторони у вирішенні існуючих недоліків ступінь впливу даних сторін на вирішення проблем. Як наслідок було запропоновано комерційне рішення з конкурентними перевагами, що задовольняє інтереси зацікавлених осіб,, та виділено унікальну ціннісну пропозицію запропонованого продукту. Було проведено аналіз майбутніх клієнтів досліджено сегменти ринку

споживання. Це дозволило спрогнозувати потенційні доходи та витрати на реалізацію продукту. У результаті була описана бізнес-модель, що обґрунтовує доцільність реалізації даного продукту та прогнозує його потенційну окупаємість та прибутковість в подальшому.

Таблиця 5

Канва бізнес-моделі

<table>
<tr><th>Проблема</th><th>Рішення</th><th>Унікальна ціннісна пропозиція</th><th>Прихована перевага</th><th>Споживачі</th></tr>
<tr><td rowspan="2">Відсутність цілісного рішення;

Висока вартість людських послуг;

Недоскональ-ність сучасних алгоритмів.</td><td>Програмне забезпечення що містить модуль зв'язку з клієнтами, а також модуль для аналізу користуваць - кої статистики.</td><td rowspan="2">Аналіз користува-цької активності по кожному елементу інтерфейсу.</td><td>Врахування значення важливості кожного елементу інтерфейсу.</td><td rowspan="2">Компанії, що реалізову-ють свої послуги за допомогою веб-сайтів.</td></tr>
<tr><td>Ключові метрики</td><td>Канали збуту</td></tr>
<tr><td colspan="3">Структура витрат</td><td colspan="2">Потоки доходів</td></tr>
<tr><td colspan="3">Утримання персоналу для надання технічної підтримки (виплат заробітних плат, соціальних виплат); утримання робочих місць персоналу.</td><td colspan="2">Доходи від продажу ліцензій.</td></tr>
</table>

ВИСНОВКИ

У даній магістерській роботі виконані наступні завдання:

1. Проведено аналіз існуючих методів аналізу користувацьких дій в різних в сферах діяльності. Було виділено їх переваги та недоліки, і сформульовано основні критерії до розроблюваного метода на проаналізованої інформації.
2. Визначено основні аналоги веб-сервісів, що існують на сьогодні, проаналізовано їх особливості та недоліки.
3. Сформульовано новий програмний метод оцінки якості побудови інтерфейсу веб-додатку на основі аналізу користувацьких дій.
4. Реалізовано програмне забезпечення, що дозволяє забирати, оброблювати, аналізувати та подавати у читабельному вигляді інформації по зібраній статистиці користувацьких дій.
5. Проведено порівняльний аналіз ефективності сформульованого методу в порівнянні з існуючими аналогами та виконано оцінку отриманих результатів.
6. Сформульовано шляхи подальшого удосконалення запропонованого методу.

Результати, отримані при реалізації методу, показали, що запропонований метод дає кращі результати в зазначених характеристиках, а саме в тому, що середній час користувацької сесії в середньому серед протестованих 5 веб-сайтів збільшилась на 23%, кількість відвіданих сторінок збільшилось на 15%, drop rate знизився на 11%.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Guowang Miao, Jens Zander, Ki Won Sung, and Ben Slimane, Fundamentals of Mobile Data Networks, Cambridge University Press, ISBN 1107143217, 2016.
2. Perkins, C.E. Ad hoc On-demand Distance Vector (AODV) Routing / C.E. Perkins, C.E. Beldong-Royer // RFC 3561. – July 2003.
3. Broch, J.A performance comparation of multihop wireless ad hoc network routing protocols / J. Brocj, D.A. Maltz // Proc. Of MOBICOM`98 -1998.
4. Basagni, S. Mobility – Adaptive Protocols for Managing Large Ad Hoc Networks / S. Basagni. D. Turgut // Proceedings of the IEEE International Conference on Communication (ICC) – 2001 – p. 1539-1543.
5. Kawadia, V. System services for implementation Ad-Hoc routing: Architecture. Implementation and Experiences / V. Kawadia, Y Zhang, B, Gupta // Proceeding of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys) San Drancisko, C.A. – June 2010 – P. 99-112.
6. RFC 7541 — HPACK: Header Compression. Режим доступу https://tools.ietf.org/html/rfc7541
7. An Extensive Examination of Data Structures. Режим доступу: https://msdn.microsoft.com/en-us/library/ms379574(v=vs.80).aspx, вільний
8. Modeling mobility for vehicular ad-hoc networks Режим доступу: https://dl.acm.org/citation.cfm?id=1023892
9. Akyildiz I. F., Wang X., Wang W. Wireless mesh networks: a survey //Computer networks. – 2005. – Т. 47. – №. 4. – С. 445-487.
10. Perkins C. E. et al. Performance comparison of two on-demand routing protocols for ad hoc networks //IEEE Personal communications. – 2001. – Т. 8. – №. 1. – С. 16-28.

11. Ko Y. B., Vaidya N. H. Location-Aided Routing (LAR) in mobile ad hoc networks //Wireless networks. – 2000. – Т. 6. – №. 4. – С. 307-321.

12. Ko Y. B., Vaidya N. H. Location-Aided Routing (LAR) in mobile ad hoc networks //Wireless networks. – 2000. – Т. 6. – №. 4. – С. 307-321.

13. Daly E. M., Haahr M. Social network analysis for routing in disconnected delay-tolerant manets //Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing. – ACM, 2007. – С. 32-40.

14. Karp B., Kung H. T. GPSR: Greedy perimeter stateless routing for wireless networks //Proceedings of the 6th annual international conference on Mobile computing and networking. – ACM, 2000. – С. 243-254.

15. Intanagonwiwat C., Govindan R., Estrin D. Directed diffusion: A scalable and robust communication paradigm for sensor networks //Proceedings of the 6th annual international conference on Mobile computing and networking. – ACM, 2000. – С. 56-67.

16. Trifunovic S. et al. WiFi-Opp: ad-hoc-less opportunistic networking //Proceedings of the 6th ACM workshop on Challenged networks. – ACM, 2011. – С. 37-42.

17. Conti M. et al. Experimenting opportunistic networks with WiFi Direct //Wireless Days (WD), 2013 IFIP. – IEEE, 2013. – С. 1-6.

18. Arnaboldi V., Conti M., Delmastro F. CAMEO: A novel context-aware middleware for opportunistic mobile social networks //Pervasive and Mobile Computing. – 2014. – Т. 11. – С. 148-167.

19. Casetti C. et al. Content-centric routing in Wi-Fi direct multi-group networks //World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a. – IEEE, 2015. – С. 1-9.

20. Broch J. et al. A performance comparison of multi-hop wireless ad hoc network routing protocols //Proceedings of the 4th annual ACM/IEEE

international conference on Mobile computing and networking. – ACM, 1998. – С. 85-97.

21. Perkins C., Belding-Royer E., Das S. Ad hoc on-demand distance vector (AODV) routing. – 2003. – №. RFC 3561.

22. Lee S. J., Gerla M. AODV-BR: Backup routing in ad hoc networks //Wireless Communications and Networking Confernce, 2000. WCNC. 2000 IEEE. – IEEE, 2000. – Т. 3. – С. 1311-1316.

23. Chakeres I. D., Klein-Berndt L. AODVjr, AODV simplified //ACM SIGMOBILE Mobile Computing and Communications Review. – 2002. – Т. 6. – №. 3. – С. 100-101.

ДОДАТКИ

Додаток 1
Лістинг програми

Лістинг 1. SignalHandler.cs

```
namespace EuroDiff
{
   class Program
   {
      static void Main(string[] args)
      {
         int command = 0;
         while (true)
         {
            try
            {
               command = Convert.ToInt32(Console.ReadLine());
               break;
            }
            catch
            {
               Console.WriteLine("Invlid input, try again");
            }
         }

         List<Country> countryList = new List<Country>();
         List<City> cityList = new List<City>();
         for (int i = 0; i < command; i++)
         {
            string[] countryInput = Console.ReadLine().Split(' ');
            string countryName = countryInput[0];

            int xl = 0;
            try
            {
               xl = int.Parse(countryInput[1]);
               if (xl <= 0)
               {
                  Console.WriteLine("Invalid input");
                  i--;
                  continue;
               }
            }
            catch
            {
               Console.WriteLine("Invalid input");
               i--;
               continue;
            }
            int yl = 0;
            try
            {
               yl = int.Parse(countryInput[2]);
               if (yl <= 0)
               {
                  Console.WriteLine("Invalid input");
                  i--;
                  continue;
               }
            }
```

Лістинг 1. Продовження

```
 {
    Console.WriteLine("Invalid input");
    i--;
    continue;
  }
  int xh = 0;
  try
  {
    xh = int.Parse(countryInput[3]);
    if (xh <= 0)
    {
      Console.WriteLine("Invalid input");
      i--;
      continue;
    }
  }
  catch
  {
    Console.WriteLine("Invalid input");
    i--;
    continue;
  }
  int yh = 0;
  try
  {
    yh = int.Parse(countryInput[4]);
    if (yh <= 0)
    {
      Console.WriteLine("Invalid input");
      i--;
      continue;
    }
  }
  catch
  {
    Console.WriteLine("Invalid input");
    i--;
    continue;
  }
  countryList.Add(new Country(countryName, xl, yl, xh, yh));
  for (int x = xl; x <= xh; x++)
  {
    for (int y = yl; y <= yh; y++)
    {
      cityList.Add(new City(countryName, x, y, cityList.Count));
      countryList[i].CitiesIds.Add(cityList.Count - 1);
    }
  }
}
foreach (City city in cityList)
{
  AddNeighborCity(cityList, 1, 0, city);
  AddNeighborCity(cityList, -1, 0, city);
  AddNeighborCity(cityList, 0, -1, city);
  AddNeighborCity(cityList, 0, 1, city);
  foreach (Country country in countryList)
```

Лістинг 1. Продовження

```
 {
        if (!city.coins.ContainsKey(country.name))
        {
            city.coins.Add(country.name, 0);
        }
    }
}
int borderCnt = 0;
foreach(City city in cityList)
{
    foreach(KeyValuePair<string, int> neighbor in city.neighborCities)
    {
        if(city.country != cityList[neighbor.Value].country)
        {
            borderCnt++;
        }
    }
}
if(borderCnt == 0)
{
    Console.WriteLine("Not all countries have neighbors. Invalid input.");
    Console.ReadLine();
    return;
}
int days = 1;
while (true)
{
    foreach (City city in cityList)
    {
        city.MoveCoins(countryList, cityList, city.country, city.id);
    }
    foreach (City city in cityList)
    {
        city.Finalise(countryList, cityList, city.country, city.id);
    }
    int completedCountries = 0;
    foreach (Country country in countryList)
    {
        if (country.IsComplete || country.CheckCities(cityList, countryList, days))
        {
            completedCountries += 1;
        }
    }
    if (completedCountries == countryList.Count)
    {
        foreach (Country country in countryList)
        {
            Console.WriteLine(country.name + " " + country.days);
        }
        Console.ReadLine();
        return;
    }

    days++;
}
```

Лістинг 1. Продовження

```
        private static void AddNeighborCity(List<City> cityList, int deltaX, int deltaY, City currentCity)
        {
            City tmpCity = cityList.Find(c => c.x == currentCity.x + deltaX && c.y == currentCity.y + deltaY);
            if (tmpCity != null)
            {
                if (!currentCity.neighborCities.Contains(new KeyValuePair<string, int>(tmpCity.country, tmpCity.id)))
                {
                    currentCity.neighborCities.Add(new KeyValuePair<string, int>(tmpCity.country, tmpCity.id));
                }
            }
        }
    }
}
```

Лістинг 2. ActionSet.cpp

```
#include "mongo/platform/basic.h"

#include "mongo/db/auth/action_set.h"

#include <bitset>
#include <string>

#include "mongo/base/status.h"
#include "mongo/bson/util/builder.h"
#include "mongo/util/log.h"
#include "mongo/util/str.h"

namespace AuthHandling {

ActionSet::ActionSet(std::initializer_list<ActionType> actions) {
   for (auto& action : actions) {
      addAction(action);
   }
}

void ActionSet::addAction(const ActionType& action) {
   if (action == ActionType::anyAction) {
      addAllActions();
      return;
   }
   _actions.set(action.getIdentifier(), true);
}

void ActionSet::addAllActionsFromSet(const ActionSet& actions) {
   if (actions.contains(ActionType::anyAction)) {
      addAllActions();
      return;
   }
   _actions |= actions._actions;
}

void ActionSet::addAllActions() {
   _actions = ~std::bitset<ActionType::NUM_ACTION_TYPES>();
}

void ActionSet::removeAction(const ActionType& action) {
   _actions.set(action.getIdentifier(), false);
   _actions.set(ActionType::anyAction.getIdentifier(), false);
}

void ActionSet::removeAllActionsFromSet(const ActionSet& other) {
   _actions &= ~other._actions;
   if (!other.empty()) {
      _actions.set(ActionType::anyAction.getIdentifier(), false);
   }
}

void ActionSet::removeAllActions() {
   _actions = std::bitset<ActionType::NUM_ACTION_TYPES>();
```

Лістинг 2. Продовження

```
bool ActionSet::contains(const ActionType& action) const {
   return _actions[action.getIdentifier()];
}

bool ActionSet::isSupersetOf(const ActionSet& other) const {
   return (_actions & other._actions) == other._actions;
}

Status ActionSet::parseActionSetFromString(const std::string& actionsString, ActionSet* result) {
   std::vector<std::string> actionsList;
   str::splitStringDelim(actionsString, &actionsList, ',');
   std::vector<std::string> unrecognizedActions;
   Status status = parseActionSetFromStringVector(actionsList, result, &unrecognizedActions);
   invariant(status);
   if (unrecognizedActions.empty()) {
      return Status::OK();
   }
   std::string unrecognizedActionsString;
   str::joinStringDelim(unrecognizedActions, &unrecognizedActionsString, ',');
   return Status(ErrorCodes::FailedToParse,
                 str::stream() << "Unrecognized action privilege strings: "
                               << unrecognizedActionsString);
}

Status ActionSet::parseActionSetFromStringVector(const std::vector<std::string>& actionsVector,
                                                 ActionSet* result,
                                                 std::vector<std::string>* unrecognizedActions) {
   result->removeAllActions();
   for (size_t i = 0; i < actionsVector.size(); i++) {
      ActionType action;
      Status status = ActionType::parseActionFromString(actionsVector[i], &action);
      if (status == ErrorCodes::FailedToParse) {
         unrecognizedActions->push_back(actionsVector[i]);
      } else {
         invariant(status);
         if (action == ActionType::anyAction) {
            result->addAllActions();
            return Status::OK();
         }
         result->addAction(action);
      }
   }
   return Status::OK();
}

std::string ActionSet::toString() const {
   if (contains(ActionType::anyAction)) {
      return ActionType::anyAction.toString();
   }
   StringBuilder str;
   bool addedOne = false;
   for (int i = 0; i < ActionType::actionTypeEndValue; i++) {
      ActionType action(i);
```

Лістинг 2. Продовження

```
 if (contains(action)) {
          if (addedOne) {
             str << ","

          str << ActionType::actionToString(action);
          addedOne = true;
       }
    }
    return str.str();
}

std::vector<std::string> ActionSet::getActionsAsStrings() const {
    std::vector<std::string> result;
    if (contains(ActionType::anyAction)) {
       result.push_back(ActionType::anyAction.toString());
       return result;
    }
    for (int i = 0; i < ActionType::actionTypeEndValue; i++) {
       ActionType action(i);
       if (contains(action)) {
          result.push_back(ActionType::actionToString(action));
       }
    }
    return result;
}
```

Додаток 2
Копія презентації

Програмний метод оптимізації та формалізації процесу розроблення інтерфейсів для веб-застосунків

Доповідач: студент групи КП-81мп
Царіков Максим Сергійович
Науковий керівник: к.т.н., доцент
Олещенко Любов Михайлівна

Актуальність дослідження

- Популярність веб-застосунків.
- Висока конкуренція в сфері ведення бізнесу в мережі Інтернет.
- Існування недоліків в сучасних застосунках в методах відслідковування користувацької активності.

Постановка науково-технічної задачі

Створення модуля для React.JS, що на основі користувацьких дій буде надавати повну статистику по кожному UI елементу інтерфейсу, з подальшою оцінкою ефективності його розташування.

Мета дослідження

Формалізувати процес розроблення інтерфейсу веб-застосунків, створити програмний метод, що на основі користувацької активності буде відображати недоліки в проектуванні інтерфейсу веб-застосунку.

Об’єкт та предмет досліджень

- Об’єктом дослідження є процес створення інтерфейсу для веб-застосунків.
- Предметом дослідження є методи використання часової метрики користувацьких дій для оптимізації розташування UI елементів в інтерфейсах веб-застосунків.

Технічні задачі

- Створення програмного модуля, що буде швидко встановлюватися у застосунках, що використовують React.JS.
- Створення алгоритму оцінки UI елементів на основі метрики користувацьких дій.

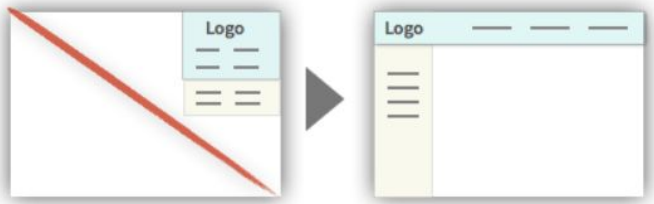
Термінологія

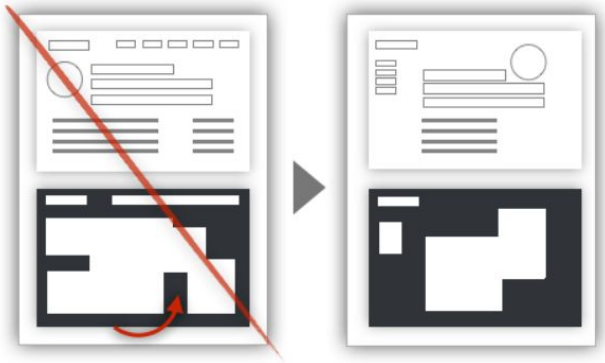
- React.JS - фреймворк для створення веб-застосунків.
- User time metrics - параметр оцінки користувацьких дій у різні проміжки часу.
- User drop rate - параметр оцінки окремих веб-сторінок, що показує величину користувачів, які припинили сесію на даному веб-застосунку

Дослідники

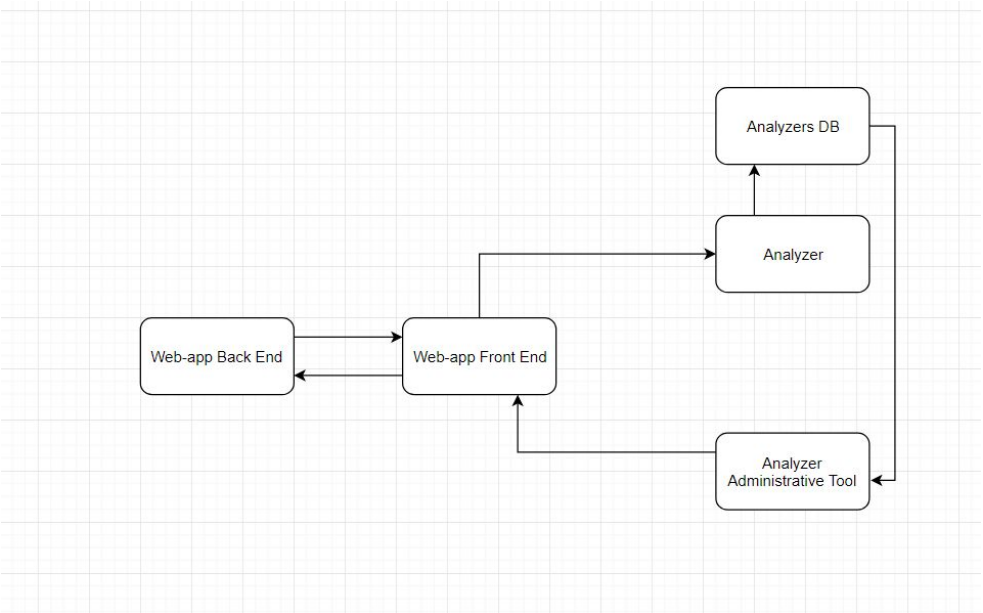
- “Research on Optimized Design of Kansei Engineering-Based Web Interface” - **Yu Chen**
- “The research on user engagement: The case of online shopping experiences” - **Elsevier B.V.**

Проблема створення та підтримки сучасних веб-застосунків

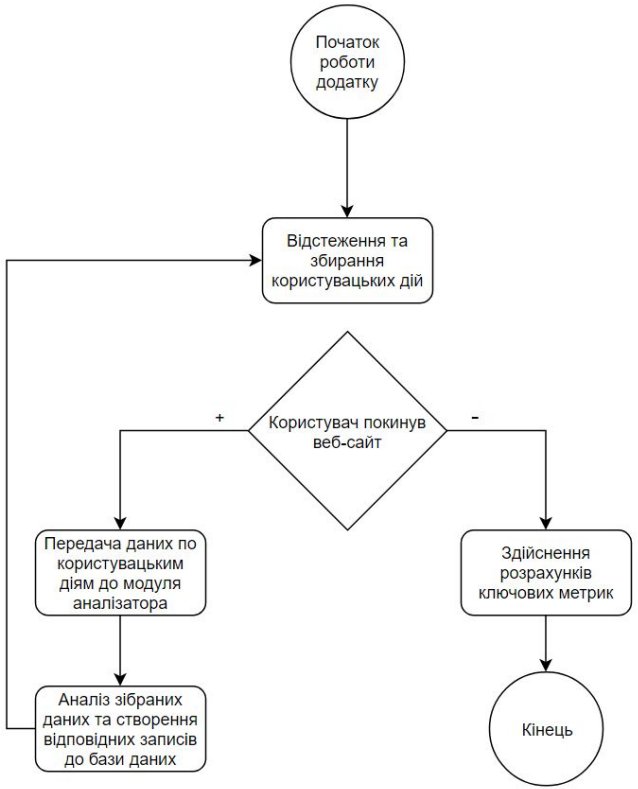




Блок схема взаємодії компонентів

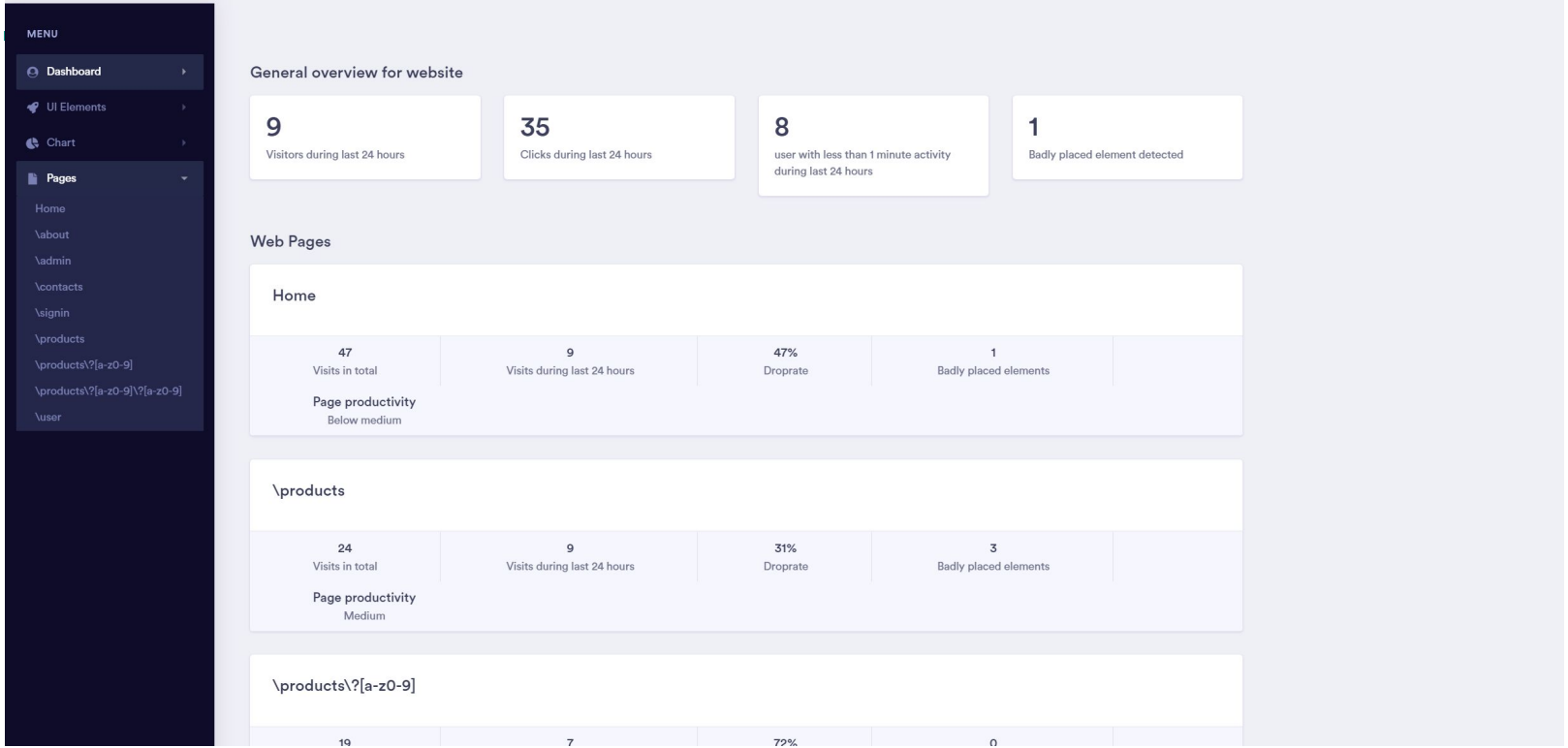


Робота алгоритму



Робота алгоритму

- Збір кількості кліків по кожному елементу інтерфейсу;
- Розбиття елементів на 2 головні групи;
- Оцінка важливості елементу в кожні групп $\frac{Ncpe}{Ncp} * 100$ (у відсотках);
- Створення таблиці, що сортується за зростанням важливості, де показано середній час пошуку елементу;
- Виділення елементів в таблиці, що погано розташовані.


MENU
Dashboard
UI Elements
Chart
Pages
Home
\about
\admin
\contacts
\signin
\products
\products\?[a-z0-9]
\products\?[a-z0-9]\?[a-z0-9]
\user
General overview for website
9
Visitors during last 24 hours
35
Clicks during last 24 hours
8
user with less than 1 minute activity during last 24 hours
1
Badly placed element detected
Web Pages
Home
47
Visits in total
9
Visits during last 24 hours
47%
Droprate
1
Badly placed elements
Page productivity
Below medium
\products
24
Visits in total
9
Visits during last 24 hours
31%
Droprate
3
Badly placed elements
Page productivity
Medium
\products\?[a-z0-9]
19
7
72%
0


MENU
Dashboard
UI Elements
Chart
Pages
Ui elements for \products
Name Overall clicks Average time for click Average clicks per session Estimated time for click
1 filter-name 34 3.72 1.02 <15
2 filter-kindOf 38 5.612 0.7 <15
3 sortBtn 24 17.782 1.1 <10
4 gridType 5 21.073 0.3 >15
Show all

Результати тестування методу

70 користувачам було запропоновано 5 веб-сайтів в 2 етапи. 1й - веб-сайт, до внесення змін за допомогою набраної статистики, другий - змінений.

- Average droprate - знизився на 15%.
- Average session duration - зросла на 23%.
- Average amount of visited pages - зросла на 14%.

Використані технології

- ReactJS;
- C++;
- MongoDB;
- git;
- HTML5, CSS;

Наукова новизна

Запропоновано новий програмний метод оцінки “важливості” елементів інтерфейсу веб-сторінки, що надає можливість формалізувати процес підтримки веб-сайтів, а також внесення змін в зовнішній вигляд веб-сторінок

<table>
<tr><td rowspan="2">Проблема

відсутність цілісного рішенння; висока вартість людських послуг; недосконаліть сучасних алгоритмів;</td><td>Рішення

програмне забезпечення що містить модуль зв’язку з клієнтами, а також модуль аналізу користувацької активності</td><td rowspan="2">Унікальна ціннісна пропозиція

аналіз користувацької активності по кожному елементу інтерфейсу кожної веб-сторінки веб-сайту</td><td>Прихована перевага

врахування значення важливості веб-сторінки для даного веб-сайту</td><td rowspan="2">Споживачі

Компанії, що реалізовують свої послуги за допомгою веб-сайтів</td></tr>
<tr><td>Ключові метрики
кількість проданих ліцензій</td><td>Канали
через відділи співпраці/інтеграції відповідних мікросервісів</td></tr>
<tr><td colspan="3">Структура витрат
утримання персоналу для надання технічної підтримки (виплати заробітних плат, соціальних виплат); утримання робочих місць для персоналу (оплата за оренду офісу та комунальні послуги);податкові витрати;</td><td colspan="2">Потоки доходів
доходи від продажу ліцензій</td></tr>
</table>

Апробація

XII наукова конференція магістрантів та аспірантів «Прикладна математика та комп’ютинг» (ПМК-2019).

Висновок

Проведено аналіз існуючих методів аналізу користувацьких дій в різних в сферах діяльності. Було виділено їх переваги та недоліки, і сформульовано основні критерії до розроблюваного метода на проаналізованої інформації.

Визначено основні аналоги веб-сервісів, що існують на сьогодні, проаналізовано їх особливості та недоліки.

Сформульовано новий програмний метод оцінки якості побудови інтерфейсу веб-додатку на основі аналізу користувацьких дій.

Реалізовано програмне забезпечення, що дозволяє забирати,, оброблювати, аналізувати та подавати у читабельному вигляді інформації по зібраній статистиці користувацьких дій.

Дякую за увагу!

Список використаних джерел

- https://www.nickkolenda.com/user-experience/
- https://themanifest.com/web-design/11-tactics-optimize-your-websites-user-experience
- https://www.awwwards.com/7-ux-tips-for-effective-conversion-rate-optimization.html
- https://vtldesign.com/what-we-do/web-strategy/user-interface-design/